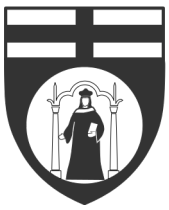


UNIVERSITÀ
DEGLI STUDI
DI GENOVA

UNIVERSITY OF GENOVA & ISTITUTO ITALIANO DI TECNOLOGIA

PHD PROGRAM IN BIOENGINEERING AND ROBOTICS

Structured Machine Learning for Robotics

by

Gian Maria Marconi

Thesis submitted for the degree of *Doctor of Philosophy* (XXXII cycle)
Curriculum in Advanced and Humanoid Robotics

December 2019

Lorenzo Rosasco	Supervisor
Giorgio Metta	Supervisor
Giorgio Cannata	Head of the PhD program

Thesis Jury:

Maurizio Filippone, *EURECOM*	External examiner
Marcello Restelli, *Politecnico di Milano*	External examiner
Massimiliano Pontil, *Istituto Italiano di Tecnologia*	Internal examiner

Dibris

Department of Informatics, Bioengineering, Robotics and Systems Engineering

Acknowledgements

The endeavor of achieving a Ph.D. is no small task, and it must be considered a shared effort more than an individual feat. And while words will never be enough to express my gratitude, one must try.
Indeed, these acknowledgments begin with my advisors: Lorenzo Rosasco and Giorgio Metta, who believed in me and supported my development as a scientist during these three years, they are the *condition sine qua non* this would not have been possible. I am also grateful to all the more experienced researchers that I have met during these three years and that fostered my intellectual growth with endless discussions, ideas, and collaborations: Alessandro Rudi, Fabio Anselmi, Guillaume Garrigos, Saverio Salzo, Silvia Villa, Stefano Vigogna, and Giulia Pasquale. Special thanks go to Carlo Ciliberto and Daniele Calandriello, the two people that taught me the most about machine learning and academia in general, after my advisors. I will never express enough gratitude to Maximilian Nickel, whose contribution to chapter four was essential. I also thank Elisa Maiettini and Luigi Carratino, with whom I've shared the difficult path of doctoral studies. Thanks to Vanessa D'Amario, Veronica Tozzo and Laura Di Rocco for the never-ending encouragement and comfort. Thanks to Raffaello Camoriano and Aiko Dinale for the variety and the quality of time spent inside and outside the laboratory.
Many technical issues would have never been overcome without the experienced advice of Daniele Domenichelli, Nicolò Genesio and, especially, Andrea Ruzzenenti, who above all showed me that it is not necessary to have a diploma to be a true engineer. An obligatory mention goes to Giulia D'Angelo, Fabrizio Bottarel and Massimiliano Iacono for being another reason for leaving my desk at the University and visit IIT.
I also want to especially acknowledge the reviewers of this thesis: Maurizio Filippone and Francesco Nori. Without their invaluable comments and feedback this thesis as it is now, would not have been possible.
I warmly thank Gianluca Alloisio, Marco Bandino, Marco Stanizzi, Alessio Corrado, Andrea Mussi, Gianluca Garzarelli Doria and Luca Demetrio for defeating the stereotype of Liguria

as an unwelcoming region, and especially Damiano Malafronte for dramatically increasing my survival chances as a Ph.D. student in Genoa.
Last but not least, I would like to thank my family and Marta for supporting my research career in the past and in the future.

Abstract

Machine Learning has become the essential tool for automating tasks that consist in predicting the output associated to a certain input. However, many modern algorithms are mainly developed for the simple cases of classification and regression. Structured prediction is the field concerned with predicting outputs consisting of complex objects such as graphs, orientations or sequences. While these objects are often of practical interest, they do not have many of the mathematical properties that allow to design principled and computationally feasible algorithms with traditional techniques.

In this thesis, we investigate and develop algorithms for learning manifold-valued functions in the context of structured prediction. Differentiable manifolds are a mathematical abstraction used in many domains to describe sets with continuous constraints and non-Euclidean geometric properties. By taking a structured prediction approach we show how to define statistically consistent estimators for predicting elements of a manifold, in constrast to traditional structured prediction algorithms that are restricted to output sets with finite cardinality. We introduce a wide range of applications that leverage manifolds structures. Above all, we study the case of the hyperbolic manifold, a space suited for representing hierarchical data. By representing supervised datasets within hyperbolic space we show how it is possible to invent new concepts in a previously known hierarchy and show promising results in hierarchical classification.

We also study how modern structured approaches can help with practical robotics tasks, either improving performances in behavioural pipelines or showing more robust predictions for constrained tasks. Specifically, we show how structured prediction can be used to tackle inverse kinematics problems of redundant robots, accounting for the constraints of the robotic joints. We also consider the task of biological motion detection and show that by leveraging the sequence structure of video streams we significantly reduce the latency of the application. Our studies are complemented by empirical evaluations on both synthetic and real data.

Publications

Some ideas, figures and tables have appeared previously in the following publications.

Rudi, A., Ciliberto, C., Marconi, G. M., & Rosasco, L. (2018). Manifold structured prediction. *In Advances in Neural Information Processing Systems (pp. 5610-5621)*.

Marconi, G. M., Rosasco, L., & Ciliberto, C. (2020). Hyperbolic Manifold Regression. *To appear in Proceedings of the 23th International Conference on Artificial Intelligence and Statistics*.

Table of contents

List of figures

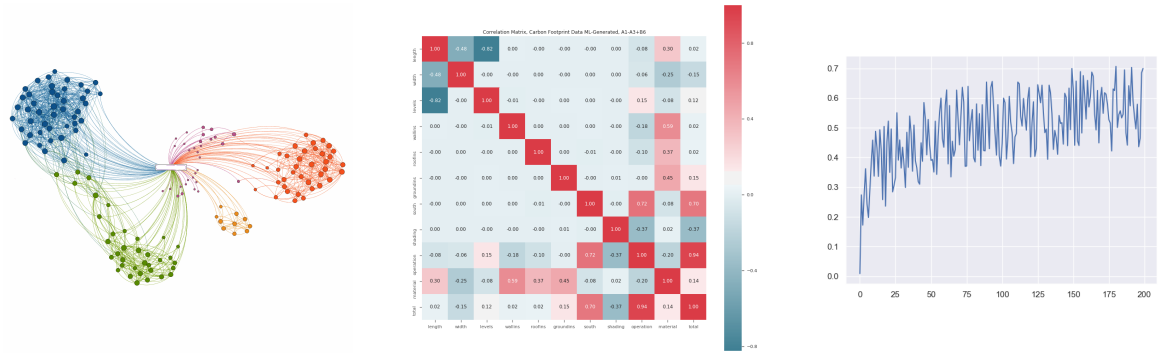
List of tables

Notation

$\mathbb{R}^d$	Euclidean space
$\mathcal{M}$	Differentiable manifold
w	column vector (if not differently specified)
$w^{(i)}$	i-th element of vector w
W	matrix
$z^\top, Z^\top$	transpose of vector z or matrix Z
$x \in \mathcal{X}$	input sample and input space
X	input matrix (each row is a sample $x^\top$)
$y \in \mathcal{Y}$	output and output space
$\hat{Y}$	output matrix (each row is an output label — or vector — $y^\top$)
$\langle \cdot, \cdot \rangle_{\mathcal{H}}$	inner product in space $\mathcal{H}$
$\| \cdot \|_p$	Euclidean p-norm, with $p = 2$ if not specified
$k(\cdot, \cdot)$	kernel function
K	empirical kernel matrix of size $n \times n$ associated to K
$\phi(\cdot)$	feature map
$\psi(\cdot, \cdot)$	structured feature map
$\psi(\cdot)$	structured output embedding
$\sigma(\cdot)$	element-wise non-linear function
I_n	$n \times n$ identity matrix
$\mathbf{0}$	zero vector
Δ	Structured Loss defined on some output space $\mathcal{Y}$
$\rho(\cdot)$	probability distribution
$\text{sign}(a)$	sign of a
∇	gradient
$\nabla_{\mathcal{M}}$	Riemannian gradient on manifold $\mathcal{M}$
$d(\cdot, \cdot)$	geodesic distance
$\Pr(\cdot)$	probability of an event

Introduction

The volume of data gathered by information system has increased of several orders of magnitude in the last decade. While data is the new primary raw material harvested by tech companies, machine learning is the new digital factory, transforming petabytes of information into digital solutions for tasks that until recently only human labour could perform. However, while the overabundance of data is one of the main reasons of modern machine learning success, as of today, the most common applications of machine learning fall into one of two categories: classification or regression. These correspond to either predicting a single number or choosing a label to classify an observed item. While these two base tasks are surprisingly useful for a wide range of more complex applications, many technological disciplines increasingly require to deal with richer, more structured domains. Examples of such output domains, where the data cannot be represented as a single scalar, are common in robotics: orientation vectors, joint configurations, behavior trees and trajectories (Calinon, 2018; Colledanchise and Ögren, 2018; Spong et al., 2006) However, while traditional scalar regression and classification have been object of in-depth studies producing scalable algorithms with theoretical guarantees (Bauer et al., 2007; Calandriello et al., 2019; Steinwart and Christmann, 2008), this has not happened for more exotic domains. This is partially due to the flexibility of these algorithms. Many structured problems can be approached with a regression (or classification) algorithm acting on a proxy of the real desired output. However, this typically comes with a set of limitations and ad-hoc fixes. Indeed, a gap still remains when working with structured domains. The aim of this thesis is to contribute towards bridging that gap, starting from a principled formalization of the problem and then moving towards practical applications in the field of robotics.



(a) Social graph (b) Covariance matrix (c) Time-series

Figure 1 Example of structured data. (a) Social graph. (b) Covariance matrix. (c) Time series.

Machine Learning

Nowadays, machine learning is the fundamental tool in many Artificial Intelligence (AI) tasks. While there are various subdomains, this thesis is focused on supervised learning. At its core, supervised learning consists in finding a mathematical function $f : \mathcal{X} \to \mathcal{Y}$ (called model) that approximates an ideal input-output relation by looking at only a finite set of input-output couples $\{x_i, y_i\}_{i=1}^{n}$. The process of finding this function is often called *learning*, as it conceptually shows similarity to the idea of humans learning from examples. If the function is *learned* following proper algorithms, it can be expected to predict accurate outputs when presented with new inputs; this property is called *generalization*. Whenever the output is not a real number or a label from a finite set, we speak of structured prediction and structured data. This case is increasingly common in modern data analysis, where the phenomena of interest are often described by complex mathematical objects such as time series in finance (Tsay, 2014), graphs in social sciences (Farasat et al., 2015) or covariance matrices in computer vision (Minh and Murino, 2017) (see Figure 1). Some of these data types can be approached by repurposing regression or classification functions, *e.g.* time-series prediction can be approximated element-wise with a regression. However, it is not clear how much performance is lost when ignoring the actual structure of the data. On the other hand, it is not always possible to find an approximation for a structured domain. Therefore, we might require a new set of mathematical machinery for solving the problem. As an example consider the case for dense structured output spaces, where most traditional structured approaches are not suited for the problem since designed for countable output spaces.

Contributions First, we address the theoretical and algorithmic issues in Chapter 3, by developing and analyzing a novel structured approach for manifold-valued regression. Differentiable manifolds are set locally isomorph to Euclidean space that lack a global vector structure. They are often used to represent sets of complex objects such as surfaces or positive definite matrices. For example, they have been extensively used for modelling key quantities in robotics dynamics and kinematics (Park and Kim, 1998; Selig, 2013; Spong, 1992; Spong et al., 2006). We show how it is possible to compute statistically consistent estimators for differentiable manifolds and how this approach leads to multiple applications with strong generalization properties on various tasks. In Chapter 4, we further exploit these results by introducing hyperbolic data representation for hierarchical datasets. We leverage the hierarchical structure that comes with many classification datasets and use recent results on hyperbolic embeddings to recast a classification problem as a manifold-valued regression problem. We highlight the advantages of using the geometric structure of hyperbolic manifolds and introduce a new application called "*taxonomy expansion*".

Robotics

Robotics is an extremely wide and diversified field whose applications range from bi-pedal locomotion to environment exploration (Thrun et al., 2002; Westervelt et al., 2018). In this context, machine learning founds a thriving environment. The idea of substituting deterministically written algorithms with example-learned models trades the cost of modeling and analyzing the system with the cost of collecting high quality data. However, the advantage of the machine learning paradigm lies in its increased generality. When an algorithm performs accurately on a task, it is expected to show a comparable performance on similar tasks. And, indeed, machine learning has enabled many succesful robotics applications in recent years (Chua et al., 2018; D'Souza et al., 2001; Maiettini et al., 2018; Pelossof et al., 2004). However, even more than other technological domains, robotics can take advantage learning algorithms for structured data (Chen and Jackson, 2011; Pucci et al., 2016; Tosun et al., 2014), see Figure 2 for the examples of sphere and positive definite matrices. This advantages include reducing the amount of data needed for training a model, having theoretical guarantees on the algorithm performance or reducing the computational complexity for real-time applications.

Contributions In this thesis we propose two applications of structured prediction to improve the perfomance on robotic tasks. In Chapter 5, we consider the problem of learning the inverse kinematics of a robot. This problem comes with constraints in the

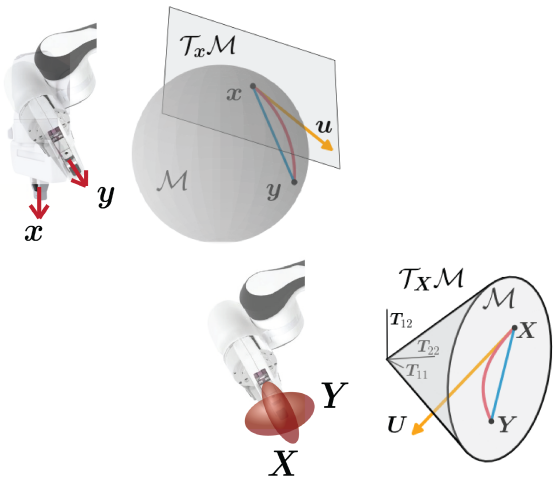


Figure 2 The sphere manifold for representing robotic end effector orientation and the cone of positive definite matrices for stiffness matrices

output space that correspond to the joint rotation boundaries. We compare our approach with another structured approach and an unstructured regression. Then, we compare them on the task of trajectory reconstruction, showing how our approach is faster to train, has better accuracy and computes solutions compliant of the boundaries of the output space. In Chapter 6, we consider the problem of biological motion detection, which aims at determining if the movement present in a video stream is generated either by a biological or non-biological entity. We leverage the natural sequence-like structure of the problem and use Gated Recurrent Units networks. We compare our approach with the state of the art unstructured approach and show that it is possible to obtain a faster classification algorithm that foregoes much of the pre-processing needed in the state of the art approach.

Structure of the thesis

This thesis is organized in the followig way. Part I is made by two introductive chapters. In Chapter 1, we introduce the fundamental ideas of machine learning, focusing on supervised learning. We introduce the definition of statistical learning and a first example algorithm called Regularized Least Square. We then introduce the concept of feature maps and kernels, showing how simple algorithms can easily be extended with these concepts. In Chapter 2, we propose a formalization of some common algorithms for structured prediction, we introduce the model and the minimization problem while

highlighting their main characteristics from a statistical and computational perspective. Part II focuses on structured prediction for manifold-valued functions. In Chapter 3, we introduce a structured estimator for learning manifold-valued functions. We prove it is statistically consistent and validate it on multiple datasets defined over different type of manifolds. In Chapter 4, we leverage the manifold-structured estimator to introduce hyperbolic data representation: a manifold valued embedding of supervised datasets with hierarchical structure. We show how to compute this representation and how to apply it to tasks of hierarchical classification using a manifold-structured estimator. Furthermore, we introduce the task of taxonomy expansion in which we are able to reconstruct a taxonomy embedded in hyperblic space. Part III revolves around two structured robotics applications. In Chapter 5, we propose an improved structured algorithm for learning the inverse kinematics of robotic arms, comparing it with both structured and unstructured algorithms. In Chapter 6, we consider the problem of biological motion detection and show how to obtain faster pipeline when tackling the problem with a structured algorithm based on deep learning. We conclude the thesis with Section 6.4 where we wrap up the results of this work.

Part I

Background

Chapter 1

Statistical Learning

In this chapter, we introduce what machine learning is about. While there are many tasks that can be tackled within this framework, given the scope of this thesis we will focus on supervised learning. We start by introducing the problem of statistical learning and the main related concepts. Theoretical discussions are followed by the example of one of the simplest algorithms for regression and binary classification: regularized least squares. The aim of this chapter is to define the fundamental mathematical tools of supervised learning while establishing a common notation. This will pave the road for more involved algorithms used in the field of structured prediction.
The rest of this chapter is organized as follows. First, we define the problem of supervised learning in the statistical sense. Then we introduce regularized least squares as a prototype algorithm. We proceed to introduce feature maps and kernel models and finally we show how to extend regularized least squares with kernels.

1.1 Supervised Learning

The goal of supervised learning is to find an input-output relation based on a finite set of input-output couples $\mathcal{D} = \{x_i, y_i\}_{i=1}^n$, called *training set*. Mathematically, this problem can be seen as finding a function $\hat{f}\colon \mathcal{X} \to \mathcal{Y}$ where $x_i \in \mathcal{X},\ y_i \in \mathcal{Y} \quad \forall\ i = 1, \dots, n$, such that given a new input x_{test}, $\hat{f}(x_{\text{test}})$ is a good estimate (or prediction) of the corresponding output y_{test}. The process of finding this function is also called "learning".
If we assume that there exists some joint distribution $\rho\colon \mathcal{X} \times \mathcal{Y} \to [0,1]$ between input and output sets, it is clear that just computing a function interpolating the dataset points might not be enough: there might be different values of y_i corresponding to a same x_i. To

model this problem we introduce the *expected risk*:

$$\mathcal{E}(f) = \int_{\mathcal{X}\times\mathcal{Y}} \Delta\big(y, f(x)\big)\, d\rho(x,y). \tag{1.1}$$

Given a *loss* function $\Delta\colon \mathcal{Y}\times\mathcal{Y} \to \mathbb{R}$ measuring the quality of a prediction $\hat{f}(x_{\text{test}})$ with respect to y_{test}, the expected risk gives an average estimate of how much a function f is approximating the stochastic relation between x and y.
Thus, the problem can be written as

$$f^* = \arg\min_{f\in\mathcal{F}} \mathcal{E}(f), \tag{1.2}$$

where $\mathcal{F}$ is the space of functions such that Equation (1.2) is well defined.
Because the expected risk is the average loss with respect to the input-output distribution, we expect that, on average, a function minimizing Equation (1.1) will give good estimates on new test points. In this case, we say that the function *generalizes* to unseen data.

1.2 Data and structure

We defined the training set as the set of input-output pairs used to learn from, with $\mathcal{X}$ the input space and $\mathcal{Y}$ the output space. We call $Z = \mathcal{X}\times\mathcal{Y}$ the data space. We introduce some notable examples of input domains that are commonly used in machine learning.

- *Euclidean space*, $\mathcal{X} = \mathbb{R}^d$, $x = [x^{(1)}, \dots, x^{(d)}]$. This is the simplest and most common input space, it is a common abstraction for many problems
- *Probability distributions.* It is possible to define $\mathcal{X} = \{x \in \mathbb{R}^d_+ : \sum_{j=1}^d x^{(j)} = 1\}$, *i.e.* the probability simplex. Given a finite set Ω of cardinality d, we can interpret each element of $\mathcal{X}$ as a probability mass function defined on the elements of Ω
- Given a graph $G = \{V, E\}$ where V are nodes of the graph and E the edges connecting such nodes, either set can be seen as an input space
- *Strings.* For a finite alphabet Σ of symbols, it is possible to consider $\mathcal{X} = \Sigma^p$, $p \in \mathbb{N}$ which is the finite space of all possible combinations of p symbols

This list is just a glimpse of the many possible domains that can be used in machine learning. As we will show later, it is easy to compute vectorial representations of the most exotic data and thus fall in the case of Euclidean space as input space. While the theory

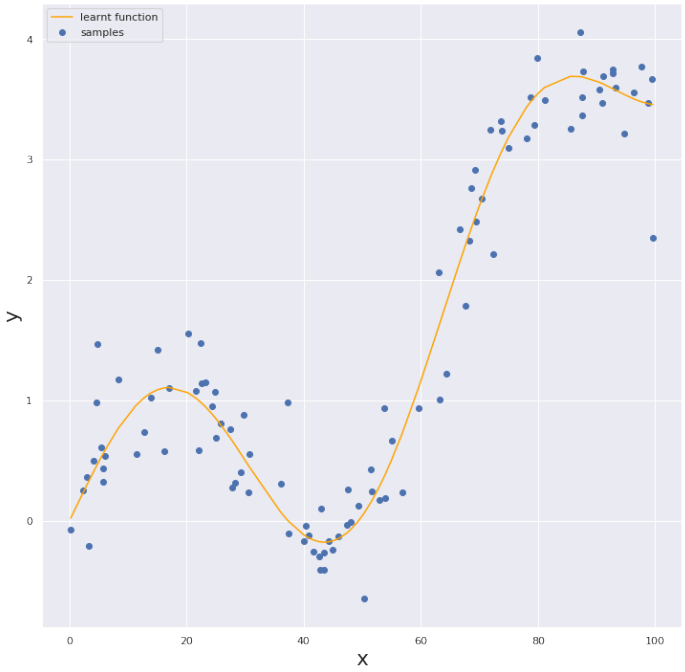


Figure 1.1 An example of regression function on a 1-dimensional training set.

we will introduce smoothly generalizes to a wide range of input spaces, the same does not hold for more exotic output spaces. The two most common and studied output domains lead to the applications known as *regression* (*i.e.* $\mathcal{Y} \subseteq \mathbb{R}$) and *binary classification* (*i.e.* $\mathcal{Y} = \{-1, 1\}$).
Note that in both cases, the problem reduces to learning a function $\hat{f} \colon \mathcal{X} \to \mathbb{R}$, where in the case of binary classification, the algorithm uses the sign of such function to classify inputs into two different classes.
By working with real-valued functions, the mathematical machinery at our disposal is more powerful and refined, as an example think of differentiability or convexity, concepts well defined for real-valued functions. However, it is often the case that output spaces are comprised of more exotic elements with a more interesting structure. Some of these spaces are:

- *Vector spaces*, $\mathcal{Y} = \mathbb{R}^T$, which is the problem of multivariate regression
- *Multilabel sets*, $\mathcal{Y} = 2^{\{1,2,\dots,T\}}$, $T \in \mathbb{N}$, each output is any subset of T categories
- *Nodes of a graph*, $\mathcal{Y} = V$ where V are the nodes of a graph $G = \{V, E\}$
- *Space of distributions*, $\mathcal{Y}$ is a space of measures

Studying how to effectively learn functions with such output spaces might be challenging, and that is the focus of structured machine learning.

1.2.1 Loss functions

We have introduced the expected risk in Equation (1.1), as a functional parametrized by a *loss function*, a function that quantifies the cost we incur in predicting $\hat{f}(x_{\text{test}})$ when the likely answer is y_{test}. Because the learning process involves a minimization problem, the type of loss dictates the difficulty of the problem itself and thus it is intimately related to the learning algorithm used to train the model. Since we mentioned that the two most common supervised machine learning tasks are regression and binary classification, we introduce some common losses for these two. Examples for regression are the square loss $\Delta_2 \colon \mathbb{R}\times\mathbb{R} \to [0,\infty), \quad (y,z) \mapsto (y-z)^2$; the absolute loss: $\Delta_1 \colon \mathbb{R}\times\mathbb{R} \to [0,\infty), \quad (y,z) \mapsto |y-z|$, and the hinge loss $\Delta_\epsilon \colon \mathbb{R}\times\mathbb{R} \to [0,\infty), \quad (y,z) \mapsto \max_\epsilon\{\epsilon, |y-z|\}$. For classification, a widely used loss is the misclassification loss function $\Delta_b \colon \mathbb{R}\times\{-1,+1\} \to \{0,1\}, \quad (y,z) \mapsto \mathbb{1}_{\text{sgn}(y)\neq z}$ where $\mathbb{1}\colon \mathbb{R} \to \{0,1\}$ is the indicator function. These examples are strictly tailored for the tasks of regression or classification, this implies that the output data belongs to either the real line or a binary set. If we want to consider more structured data, then also the loss function needs to be selected based on the new space. While this question might have natural answers in some cases (*e.g.* vector spaces), in general it is not an easy problem, especially if we take into account how the complexity of the problem might change as the loss function changes. Some examples of such functions are the Euclidean distance for vector spaces, Wasserstein distance for probability distributions (Peyré et al., 2019), the Sequence-to-sequence structured losses (Edunov et al., 2017), and the Shifted inner product similarity for graph nodes (Okuno et al., 2018).
In general, when dealing with datasets where the output space $\mathcal{Y}$ does not have a clear vector structure, it is necessary to find a suitable loss function in the form

$$\Delta \colon \mathcal{Y}\times\mathcal{Y} \to [0,\infty) \tag{1.3}$$

that appropriately measures the similarity between two elements of $\mathcal{Y}$ and leads to a computationally feasible form of Equation (1.2). We will see a study of such cases in the next parts of this thesis Chapter 3.

1.3 Learning and generalization

We have introduced the problem of supervised learning from a dataset, however we have not discussed yet what does it mean to *learn* and how an algorithm can achieve such behaviour. We can think of a learning algorithm as a map $\mathcal{A}, \mathcal{D}_n \mapsto \widehat{f}$ that for each training set $\mathcal{D}_n$ returns a function $\widehat{f}$ that either solves Equation (1.2) or approximates well enough its solution. A desirable property of an algorithm is consistency.

Definition 1.3.1 (Consistency)**.** *Given a learning algorithm $\mathcal{A}, \mathcal{D}_n \mapsto \widehat{f}$ where $\mathcal{D}_n$ is a training set sampled from a fixed distribution ρ and n is the size of the training set, the learning algorithm is* ***consistent*** *for ρ iff*

$$\lim_{n\to\infty} \mathbb{P}\left(\mathcal{E}(\widehat{f}) - \inf_{f\in\mathcal{F}} \mathcal{E}(f) > \epsilon\right) = 0 \qquad \forall\, \epsilon > 0. \tag{1.4}$$

If the algorithm is consistent for any ρ then we say that it is universally consistent.

This allows us to give an operative definition of *supervised learning.*
Given a loss function and sample set of input-output pairs, i.i.d with respect to some fixed and unknown distribution, the goal of supervised learning is to find a universally consistent algorithm.
We now introduce another fundamental definition.

Definition 1.3.2 (Uniform consistency and sample complexity)**.** *Given a learning algorithm $\mathcal{A}, \mathcal{D}_n \mapsto \widehat{f}$ where $\mathcal{D}_n$ is a training set sampled from a fixed distribution ρ and n is the size of the training set, the learning algorithm is* ***uniformly universally consistent*** *iff*

$$\lim_{n\to\infty} \sup_{\rho} \mathbb{P}\left(\mathcal{E}(\widehat{f}) - \inf_{f\in\mathcal{F}} \mathcal{E}(f) > \epsilon\right) = 0 \qquad \forall\, \epsilon > 0. \tag{1.5}$$

This condition can be rewritten for a certain distribution ρ using the definition of limit. An algorithm is ***uniformly consistent*** *with respect to ρ iff $\forall\, \delta, \epsilon > 0, \exists\, n_\rho(\epsilon, \delta)$ such that, if $n \geq n_\rho(\epsilon, \delta)$, then*

$$\mathbb{P}\left(\mathcal{E}(\widehat{f}) - \inf_{f\in\mathcal{F}} \mathcal{E}(f) \geq \epsilon\right) \leq \delta. \tag{1.6}$$

Here $n_\rho(\epsilon, \delta)$ is called sample complexity *and depends on the distribution ρ.*

It can be proved that it is impossible to achieve uniform universal consistency (Whitley and Watson, 1970), this result is also known as *No Free Lunch Theorem.* We will see in the following that a possible approch to overcome this issue is to consider smaller function

spaces $\mathcal{F}_{\mathcal{H}} \subset \mathcal{F}$ called *hypothesis space*. By limiting the space of possible functions, we are effectively restricting the initial problem to an easier problem.

1.4 Empirical risk minimization

Until now we have discussed about the defining features of the learning problem and algorithms used to approach it, however we left out an important hypothesis. Consider the expected risk in Equation (1.2); in general it is assumed that the distribution $\rho(x, y)$ is unknown, as it is in most practical cases. This makes impossible to solve problem Equation (1.2) and find the minimizing f^*. Nevertheless we have access to a finite set of samples in the form of training set $\mathcal{D}_n$; to use this information we design algorithms that minimize the error on finite training sets as a proxy for the true problem. For this reason, we introduce the *empirical risk*

$$\widehat{\mathcal{E}}(f) = \frac{1}{n} \sum_{i=1}^{n} \Delta\big(y_i, f(x_i)\big). \tag{1.7}$$

This leads our actual problem to be the so-called empirical risk minimization (ERM)

$$\inf_{f \in \mathcal{F}_{\mathcal{H}}} \widehat{\mathcal{E}}(f), \tag{1.8}$$

where $\mathcal{F}_{\mathcal{H}} \subset \mathcal{F}$ is called *hypothesis space*. Ideally the hypothesis space should be complex enough that the solution of Equation (1.8) approximates well enough the solution of Equation (1.2), while still leading to an algorithm with feasible computations. A traditional approach to control the complexity of $\mathcal{F}_{\mathcal{H}}$ is regularization. Regularization introduces in the learning algorithm a scalar term that penalizes the complexity of the sought function. A regularized approach solves the problem

$$\inf_{f \in \mathcal{F}_{\mathcal{H}}} \widehat{\mathcal{E}}(f) + \lambda R(f), \tag{1.9}$$

where $\lambda \in \mathbb{R}_+$, and $R \colon \mathcal{F}_{\mathcal{H}} \to [0, \infty)$ is some functional that penalizes complexity in f.

1.5 Regularized least squares

After discussing basic properties of learning algorithms and data spaces, we are finally ready to introduce an example of learning algorithm, namely Regularized Least Squares

(RLS). RLS is a regression algorithm that learns a linear function $f_w(x) = \langle w, x \rangle$ to approximate a relation between Euclidean space $\mathbb{R}^d$ and the real line $\mathbb{R}$. It solves the following version of Equation (1.8)

$$\min_{w \in \mathbb{R}^d} \frac{1}{n} \sum_{i=1}^{n} \big(y_i - \langle w, x_i \rangle\big)^2 + \lambda \|w\|_2, \quad \lambda \geq 0. \tag{1.10}$$

We make some remarks on Equation (1.10). First, the chosen loss for this problem is the square loss $\Delta(y, z) = (y - z)^2$. Second, the regularization functional is the Euclidean norm of the vector w. Lastly, the hypothesis space is the class of linear functions

$$\mathcal{F}_{\mathcal{H}} = \big\{ f : \mathbb{R}^d \to \mathbb{R} \quad | \quad f_w(x) = \langle w, x \rangle \,,\ w \in \mathbb{R}^d \big\}.$$

An advantage of this approach is that the square loss is convex. Because the sum of a convex functions and strongly convex functions is a strongly convex function, finding any critical point of the functional is enough to find its unique minimum (Peypouquet, 2015). It is possible to express the regularized empirical risk in vectorial form

$$\frac{1}{n} \| Y - Xw \|^2 + \lambda \| w \|_2, \tag{1.11}$$

where $Y \in \mathbb{R}^n$ is a vector $Y = [y_1, \ldots, y_n]$ containing all the output points of the dataset and $X \in \mathbb{R}^{n \times d}$ is a matrix where each row is a point of the training set. The gradient with respect to w is

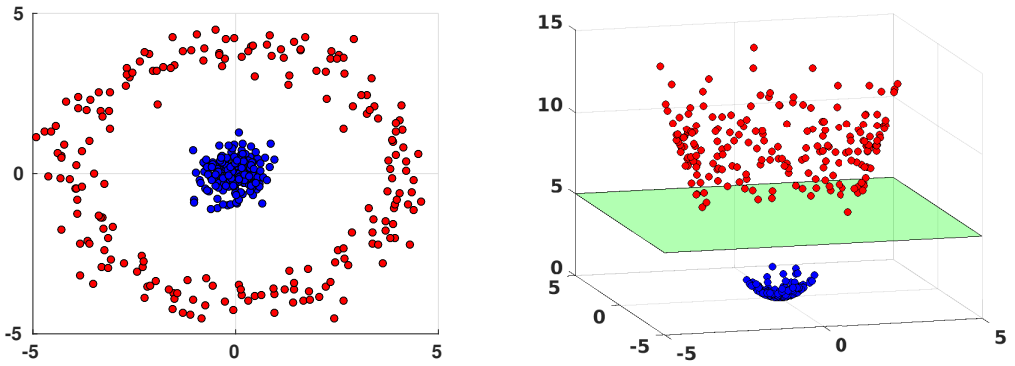
$$\nabla_w \Big(\frac{1}{n} \| Y - Xw \|^2 + \lambda \| w \|_2 \Big) = -\frac{2}{n} X^\top (Y - Xw) + 2w. \tag{1.12}$$

Because the functional is convex with respect to w, by imposing the gradient to be 0 it is possible to solve for the minimizing w in closed form

$$w = XY^\top \big(XX^\top + n\lambda I_n \big)^{-1}, \tag{1.13}$$

where $I_n \in \mathbb{R}^{n \times n}$ is the identity matrix. Note that the matrix between parentheses is always invertible because it is the sum of a positive semi-definite matrix and a positive definite matrix. Solving this linear system corresponds to training the model f_w on the training set. This algorithm can be easily adapted to the task of binary classification by changing the space of functions to be defined as

$$\mathcal{F}_{\mathcal{H}} = \big\{ f : \mathbb{R}^d \to \mathbb{R} \quad | \quad f_w(x) = \operatorname{sgn}(\langle w, x \rangle) \big\}. \tag{1.14}$$



(a) Data non linearly separable (b) Data with a new dimension introduced

Figure 1.2 An example of a non linearly separable dataset before and after applying a feature map.

Indeed, the problem of regression can be described as the task of finding a hyperplane w that approximates the data, while binary classification is the task of finding a hyperplane w that separates the data. In general, when we mention a *model*, we refer to the definition of a function with trainable parameters such as $f_w(x) = \langle w, x \rangle$.

1.6 Feature maps and Reproducing Kernel Hilbert Spaces

We devote the rest of this chapter to a powerful set of mathematical tools for defining spaces of functions. This point of view is the basis for some of the algorithm that will be used in the rest of this thesis.

1.6.1 Feature maps

Feature maps are a tool used to find suitable representations of input data such that Equation (1.8) becomes easier to solve.

Definition 1.6.1 (Feature map)**.** *Given a space of functions $\mathcal{F}_\mathcal{H}$ with vector structure, a **feature map** is a map $\phi : \mathcal{X} \to \mathcal{F}_\mathcal{H}$ such that $\langle \phi(x_1), \phi(x_2) \rangle_{\mathcal{F}_\mathcal{H}} < \infty \ \forall x_1, x_2 \in \mathcal{X}$.*
$\mathcal{F}_\mathcal{H}$ is called feature space.

To give an intuition on why feature maps are useful, consider the problem of classification depicted in Figure 1.2a. It is impossible to find a hyperplane in $\mathbb{R}^2$ (a line) that separates the two classes.

Let us define a simple feature map: $\phi : \mathbb{R}^2 \to \mathbb{R}^3,\ \phi([x_1, x_2]) = [x_1, x_2, x_1^2 + x_2^2]$. This function maps the points from the original data set into a higher dimensional space, notice that this can also be seen as a space of functions spanned by the basis: $[\ x_1\ 0\ 0\ ]^T$, $[\ 0\ x_2\ 0\ ]^T$ $[\ 0\ 0\ x_1^2 + x_2^2\ ]^T$. In figure Figure 1.2b, it can be qualitatively observed what happens after ϕ is applied to every point in the training set: the data on the external ring results in a higher x_3 coordinate making data easily separable by a hyperplane. In this example, it is sufficient to add a dimension to find a hyperplane that divides data; however, for more entangled data sets, adding many dimensions and then finding the hyperplane may require more complex feature maps.

1.6.2 Reproducing Kernel Hilbert Spaces

In Section 1.4, we showed that empirical risk minimization (Equation (1.7)) can be used as a proxy for solving expected risk minimization (Equation (1.2)). However, this problem depends on the considered hypothesis space $\mathcal{F}_{\mathcal{H}}$. We now introduce a particular class of spaces of functions that are fundamental to some classical machine learning algorithms. This class of spaces is called Reproducing Kernel Hilbert Spaces (RKHS), and it is strictly related to the notion of reproducing kernel (or positive definite function) and feature map. We start by giving the definition of kernel.

Definition 1.6.2 (Reproducing kernel)**.** *A **reproducing kernel** is a function $k : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$ that is symmetric in its arguments and such that:*

$$\sum_{i,j=1}^{M} \alpha_i \alpha_j k(x_i, x_j) \geq 0, \tag{1.15}$$

$\forall \alpha_1, \ldots, \alpha_M \in \mathbb{R}$ *and* $\forall x_1, \ldots, x_M \in \mathcal{X},\ M \in \mathbb{N}$

An important remark on the kernel matrix defined as $\{K\}_{ij} = k(x_i, x_j),\ i, j \in \{1, \ldots, M\}$ is that it is symmetric and positive semi-definite.
Every reproducing kernel identifies a space of functions $\mathcal{F}_{\mathcal{H}}$. In order to characterize it let us define a pre-Hilbert space making use of a generic k:

$$\mathcal{F}_{\mathcal{H}0} := \left\{ f : \mathcal{X} \to \mathbb{R} : f(x) = \sum_{i=1}^{M} k(x, x_i)\alpha_i \right\} \tag{1.16}$$

with $\alpha_i \in \mathbb{R}\ \forall i, \quad M \in \mathbb{N}, \quad x_i \in \mathcal{X}\ \forall i$.
We define a *Reproducing Kernel Hilbert Space* (RKHS) $\mathcal{F}_{\mathcal{H}}$ with k embedded, as the metric completion of $\mathcal{F}_{\mathcal{H}0}$.

Given two functions:

$$f(x) = \sum_{i=1}^{M} k(x, x_i)\alpha_i$$
$$g(x) = \sum_{j=1}^{M'} k(x, x_j)\beta_j$$

we can equip the space $\mathcal{F}_{\mathcal{H}}$ with an inner product.

Definition 1.6.3 (RKHS inner product)**.**

$$\langle f, g\rangle_{\mathcal{F}_{\mathcal{H}}} := \sum_{i,j}^{M,M'} k(x_i, x_j)\alpha_i\beta_j. \tag{1.17}$$

Notice that the function defined as $k_x(x') \equiv k(x', x)$ belongs to $\mathcal{F}_{\mathcal{H}}$ (just choose $M = 1$, $x_1 = x$ and $\alpha_1 = 1$). From the definition of inner product on $\mathcal{F}_{\mathcal{H}}$, we introduce a powerful property of kernels.

Theorem 1.6.1. *For a RKHS and its embedded kernel $k : \mathcal{X} \times \mathcal{X} \to \mathbb{R}^d$ we have the following property, called the reproducing property:*

$$\langle f, k_x\rangle_{\mathcal{F}_{\mathcal{H}}} = f(x), \qquad \forall f \in \mathcal{F}_{\mathcal{H}}, \forall x \in \mathcal{X}. \tag{1.18}$$

The proof can be written in a single line based on definition (1.6.3):

$$\langle f, k_x\rangle_{\mathcal{F}H} = \textstyle\sum_{i=1}^{M} k(x_i, x)\alpha_i = f(x).$$

We have shown a way to build a RKHS from a positive definite function called the kernel. It can be proved that every RKHS identifies a unique kernel and vice versa (Daumé III (2004)).

1.6.3 Feature maps and kernels

An important connection between feature maps and RKHS can be made by choosing a feature map $\phi\colon \mathcal{X} \to \mathcal{F}_{\mathcal{H}}$ such that $k_{x'}(\cdot) = \phi(x')$. By doing so, we have that $\mathcal{F} = \mathcal{F}_{\mathcal{H}}$ and the associated RKHS $\mathcal{F}_{\mathcal{H}}$ is now a space of functions induced by the feature map ϕ.

Theorem 1.6.2. *Evaluating the kernel at points x, x' is equivalent to computing the inner product of the same points in the feature space of $\phi(\cdot)$*

$$k(x, x') = \langle \phi(x), \phi(x')\rangle_{\mathcal{F}_{\mathcal{H}}}. \tag{1.19}$$

The proof follows as a single line using the reproducing property (1.18):

$$\langle \phi(x), \phi(x') \rangle_{\mathcal{F}_\mathcal{H}} = \langle k_x, k_{x'} \rangle_{\mathcal{F}_\mathcal{H}} = k(x, x')$$

So by computing the kernel $k(x, x')$ we are actually computing the inner product of two points mapped into $\mathcal{F}_\mathcal{H}$.
An example of a common kernel is the Gaussian kernel, also called radial basis function.

Definition 1.6.4. *The Gaussian kernel is a reproducing kernel* $k \colon \mathbb{R}^d \to \mathbb{R}$ *defined as*

$$k(x_1, x_2) = e^{-\frac{\|x_1 - x_2\|_2^2}{2\sigma^2}}. \tag{1.20}$$

This kernel belongs to a more general class of kernels, called translation invariant kernels, that can be characterized by their Fourier transform (Yao, 1967). The feature map associated to the Gaussian kernel maps into an infinite dimensional space whose basis can be computed from the Taylor expansion of e^x. Assuming $x \in \mathbb{R}$, we have:

$$\phi(x) = e^{-\frac{x^2}{2\sigma^2}} \left[1, \sqrt{\frac{1}{1!\,\sigma^2}}\, x, \sqrt{\frac{1}{2!\,\sigma^2}}\, x^2, \dots\right]^\top. \tag{1.21}$$

This shows that kernels may allow to define algorithms in hypothesis spaces that are otherwise computationally unfeasible such as the infinite dimensional space associated to the Gaussian kernel.

1.6.4 Kernel Regularized Least Squares

Consider the classification problem depicted in Section 1.6.1, we mentioned that it consists in finding a hyperplane (*i.e.* a linear function) in some feature space $\mathcal{F}_\mathcal{H}$ to separate the two data classes. We use the fact that a function $f \in \mathcal{F}_\mathcal{H}$ can be written as

$$f(x) = \textstyle\sum_{i=1}^{M} k(x, x_i)\, w_i$$

to show that it corresponds to a vector in the feature space spanned by ϕ:

$$\begin{aligned} f(x) &= \sum_{i=1}^{M} k(x, x_i)\, w_i \\ &= \sum_{i=1}^{M} \langle \Phi(x), \Phi(x_i) \rangle\, w_i \\ &= \langle \Phi(x), \sum_{i=1}^{M} \Phi(x_i)\, w_i \rangle \\ &= \langle \Phi(x), v \rangle. \end{aligned}$$

Thus, each function expressed as $f(x) = \sum_{i=1}^{M} k(x, x_i)\, w_i$ identifies a hyperplane $v \in \mathcal{F}_{\mathcal{H}}$. This suggests that is is possible to generalize RLS to any RKHS by simply choosing a kernel. This intuition is proved by the representer theorem.

Theorem 1.6.3 (Representer theorem)**.** *Given a training set* $\mathcal{D} = \{(x_i, y_i)\}_{i=1}^{N}$*, the solution to problem (1.8) can be expressed as*

$$\hat{f}(x) = \sum_{i=1}^{N} k(x, x_i)\, w_i, \quad \forall x \in \mathcal{X} \tag{1.22}$$

with $w \in \mathbb{R}^N$ *(Scholkopf and Smola, 2001).*

This ensures that any function in the hypothesis space can be expressed as a linear combination of kernels centered at the training points. Therefore, the problem reverts to finding the coefficients w_i for such summation. Moreover, the representer theorem ensures that the learnt function $\hat{f}$ can be expressed by a finite set of points, allowing for a representation with finite space complexity. The empirical risk minimization problem can be rewritten as

$$\min_{f \in \mathcal{F}_{\mathcal{H}}} \frac{1}{n} \sum_{i=1}^{n} \big(y_i - f(x)\big)^2 + \lambda \|f\|_{\mathcal{F}_{\mathcal{H}}}, \quad \lambda \geq 0, \tag{1.23}$$

which in vector form corresponds to

$$\min_{w \in \mathbb{R}^n} \frac{1}{n} \|\hat{Y} - Kw\|_{\mathbb{R}^n}^2 + \lambda \|f\|_{\mathcal{H}}^2, \quad \lambda \geq 0 \tag{1.24}$$

with $\hat{Y} \in \mathbb{R}^n$ a vector containing the entries of each output of the training set, $w \in \mathbb{R}^n$ the vector of weights, and $K \in \mathbb{R}^{n \times n}$ a symmetric matrix with entries $\{K\}_{ij} = k(x_i, x_j)$. Similarly to RLS, Equation (1.24) is strongly convex, therefore, computing the gradient with respect to w and setting it equal to a vector of zeros defines a linear system whose solution leads to the minimizing w. Specifically, we have

$$w = \big(n\lambda I_n + K\big)^{-1} \hat{Y}. \tag{1.25}$$

Chapter 2

Structured Prediction

Most traditional machine learning algorithms are designed for the tasks of binary classification and regression, which have as output spaces respectively the binary set $\{-1, 1\}$ and the real line $\mathbb{R}$. However many other output spaces are often of interest in practical applications. In this chapter we introduce some of the efforts that have been made to address the problem of structured prediction, *i.e.* learning functions with output spaces more complex than the binary set and the real line. In general, a structured prediction model can be seen as a composition of an encoding function $e_w \colon \mathcal{X} \to \mathcal{F}_\mathcal{H}$ and a decoding function $d_w \colon \mathcal{F}_\mathcal{H} \to \mathcal{Y}$, both of which might be parametrized by some vector w learnt from the training set. The encoding function is used to compute a representation of the input data in some Hilbert space, while the decoding function relates elements from $\mathcal{F}_\mathcal{H}$ to their corresponding output $\widehat{y} \in \mathcal{Y}$. The decoding procedure typically involves a maximization problem and in general structured estimators can be written as

$$\widehat{f}(x) = (d_w \circ e_w)(x) = \operatorname*{arg\,max}_{y \in \mathcal{Y}} h_w(y, e_w(x)) \tag{2.1}$$

where $h_w \colon \mathcal{Y} \times \mathcal{F}_\mathcal{H} \to \mathbb{R}$ is an auxiliary function and represents the dependence of the decoder with respect to w. Indeed we have $d_w(\cdot) = \arg\max_{y \in \mathcal{Y}} h_w(y, \cdot)$. A classical interpretation adopted in probabilistic graphical model is that $h_w(y, e_w(x)) = \log \rho(y|x, w) + C$, *i.e.* h_w is the likelihood of the output y given the data x and a learned parameter w, plus a constant C independent of y. However in the following we will only introduce structured support vector machines, an approach that extends some ideas of probabilistic methods with an SVM-like model, allowing for a kernelized approach of the algorithm. After that we introduce consistent regularized structured predictors, a more recent family of algorithms that enjoys statistical consistency and finite sample bounds. We close the chapter by pre-

senting neural networks for structured problems, an approach that has recently obtained outstanding results in some domains thanks to the possibility of designing ad-hoc models for the task. In all three approaches, the output space and the chosen loss define many of the characteristics of the algorithms ranging from design of the model to computational complexity.

2.1 Structured Support Vector Machine

Structured support vector machines (SSVM) are an extension of classical support vector machine algorithm (Steinwart and Christmann, 2008), that generalize the traditional max-margin model to a wider set of output domains and losses, such as multi-class classification with hierarchical loss (Cai and Hofmann, 2004), image segmentation with pixel-wise loss (Lucchi et al., 2012), and ranking with Kendall's ranking loss (Joachims, 2002). By nature SSVM are highly customizable depending on the task at hand: in this section we aim at giving an overview on its main formulations. Given the objective of minimizing the empirical risk (see Equation (1.9)), the SSVM algorithm finds a function $f\colon \mathcal{X} \to \mathcal{Y}$, that minimizes the objective under the assumption that such function can be written as

$$\widehat{f}(x) = \arg\max_{y \in \mathcal{Y}} \left\langle w, \psi(x,y) \right\rangle_{\mathcal{F}_\mathcal{H}} \tag{2.2}$$

where $w \in \mathcal{F}_\mathcal{H}$ is a learnt vector of weights and $\psi\colon \mathcal{X} \times \mathcal{Y} \to \mathcal{F}_\mathcal{H}$ is a joint feature map on both input and output. Comparing to Equation (2.1), the encoding function $e_w(x,y) = \psi(x,y)$ is fixed by design and is defined jointly on both $\mathcal{X}$ and $\mathcal{Y}$, while h_w corresponds to the linear function $\langle w, \cdot \rangle_{\mathcal{F}_\mathcal{H}}$. Using this model for empirical risk minimization leads to the following problem

$$\min_{w \in \mathcal{F}_\mathcal{H}} \frac{1}{n} \sum_{i=1}^{n} \Delta\big(y_i, f(x)\big) + \lambda \| w \|_2 \tag{2.3}$$

Because the structured loss $\Delta(y, f(x))$ is typically non-convex and piece-wise constant, solving Equation (2.3) is not a straightforward task. A possible approach is to introduce a convex upper bound to Equation (2.3) that depends on w and minimize such bound. This is motivated by the work of Zhang et al. (2004) where it is shown that this procedure is enough to obtain a consistent algorithm for binary classification. However analogous theoretical results have yet to arise for its structural counterpart.

We will now proceed to give three different reformulation of the SSVM problem for learning the parameter vector $w \in \mathcal{F}_\mathcal{H}$. The reason for this is that the first formulation is easier to

understand but leads to a problem harder to minimize. The second and third formulation correspond to the same problem in the primal and dual form, and can be solved with more efficient algorithms.

Definition 2.1.1 (Vanilla SSVM)**.** *Let* $h_w(x,y) = \langle w, \phi(x,y) \rangle_{\mathcal{F}_\mathcal{H}}$ *with* $w \in \mathcal{F}_\mathcal{H}$ *and* $\phi\colon \mathcal{X} \times \mathcal{Y} \to \mathcal{F}_\mathcal{H}$ *be the auxiliary function for some Hilbert space* $\mathcal{F}_\mathcal{H}$*. We train* w *by minimizing*

$$\min_{w \in \mathcal{F}_\mathcal{H}} \frac{1}{n} \sum_{i=1}^{n} \Delta_b(x_i, y_i, w) + \frac{\lambda}{2} \| w \|^2_{\mathcal{F}_\mathcal{H}} \tag{2.4}$$

where

$$\Delta_b(x_i, y_i, w) = \max_{y \in \mathcal{Y}} \Delta(y_i, y) - h_w(x_i, y_i) + h_w(x_i, y) \tag{2.5}$$

is a convex upper-bound for Δ*.*

Intuitively, Equation (2.5) generalizes the Hinge loss to multiple output losses, therefore solving Equation (2.4) for w can be seen as a maximum margin algorithm, in the spirit of classical support vector machines. Notice that boundedness of Δ can be quickly proved in a few steps:

$$\begin{aligned} \Delta(y, f(x)) &\leq \Delta(y, f(x)) - h_w(x, y) + h_w(x, f(x)) \\ &\leq \max_{y \in \mathcal{Y}} \Delta(y_i, y) - h_w(x, y) + h_w(x_i, y) \\ &= \Delta_b(x_i, y_i, w) \end{aligned}$$

and convexity in w is given by the fact that Δ_b is a max over affine functions of w. A possible approach for solving approximately this problem is using subgradient descent minimization (Shor, 2012), however this has a weak convergence rate of $\mathcal{O}(\frac{n}{\epsilon^2})$, *i.e.* to reduce the distance between the current w and the minimizing w^* by a factor ϵ requires $\mathcal{O}(\frac{1}{\epsilon^2})$ algorithm iterations. To overcome this issue Joachims et al. (2009) proposed to recast the minimization problem in an alternative form that allows using cutting plane minimization, a faster minimization algorithm. We proceed to introduce this formulation and its dual form in the following.

Definition 2.1.2 (One-slack variable SSVM)**.** *Let* $\xi \in \mathbb{R}$ *be an auxiliary slack variable. For any* $\lambda > 0$*, solving the problem*

$$(w^*, \xi^*) = \underset{w \in \mathcal{F}_{\mathcal{H}}, \xi \in \mathbb{R}}{\arg\min} \frac{1}{2} \| w \|^2_{\mathcal{F}_{\mathcal{H}}} + \lambda \xi \tag{2.6}$$

subject to, for all $(\bar{y}_1, \ldots, \bar{y}_n) \in \mathcal{Y} \times \ldots, \times \mathcal{Y}$,

$$\frac{1}{n} \sum_{i=1}^{n} \left[g_w(x_i, y_i) - g_w(x_i, \bar{y}_i) \right] \geq \frac{1}{n} \sum_{i=1}^{n} \Delta(y_i, \bar{y}_i) - \xi$$

$$\xi \geq 0$$

is equivalent to solving Equation (2.4)*. For a discussion on this equivalence, see the work by Joachims et al. (2009).*

This formulation can be extended to kernelized model

Definition 2.1.3 (Kernel SSVM)**.** *Let* $k((x_1, y_1), (x_2, y_2)) = \langle \phi(x_1, y_1), \phi(x_2, y_2) \rangle$*, it is possible to compute a* ***kernelized SSVM*** *defined as*

$$\widehat{f}(x) = \underset{y \in Y}{\arg\max} \sum_{i=1}^{n} \sum_{y' \in \mathcal{Y}} \alpha_{iy} \big((x_i, y'), (x, y) \big) \tag{2.7}$$

by solving for where then problem Equation (2.6) *can be written as*

$$\min_{\alpha \in \mathbb{R}^{n \times |\mathcal{Y}|}} \sum_{\substack{i=1,\ldots,n \\ y \in \mathcal{Y}}} \alpha_{iy} - \frac{1}{2} \sum_{\substack{i=1,\ldots,n \\ y \in \mathcal{Y}}} \sum_{\substack{j=1,\ldots,n \\ z \in \mathcal{Y}}} \alpha_{iy} \alpha_{jz} \hat{K}^{ij}_{yz} \tag{2.8}$$

subject to, for $i = 1, \ldots, n$ *and for all* $y \in \mathcal{Y}$*:*

$$\sum_{y \in \mathcal{Y}} \alpha_{iy} \leq \frac{\lambda}{n}, \qquad \alpha_{iy} \geq 0$$

where we with an abuse of notation we used $y, z \in \mathcal{Y}$ *to denote* $h, l = 1, \ldots, |\mathcal{Y}|$*, i.e. the set indexing all the possible elements of* $\mathcal{Y}$*. And where* $\hat{K}^{ij}_{yz} = K^{ij}_{y_i z_i} - K^{ij}_{y_i z} - K^{ij}_{y z_i} + K^{ij}_{yz}$ *with* $K^{ij}_{yz} = k((x_i, y), (x_j, z))$*. For a derivation of this algorithm, we refer the reader to the work of Nowozin et al. (2011).*

Both of these problems can be solved with a cutting-plane algorithm. This procedure iteratively seeks the most violated constraint and adjusts the set of weights w (or α) until a function that does not violate any constraint is found. This algorithm has $\mathcal{O}(\frac{n}{\epsilon})$ computational complexity for the linear kernel, and $\mathcal{O}(\frac{n^2}{\epsilon})$ for nonlinear kernels, improving on the

sub-gradient algorithm. For a detailed definition of the algorithm and the cost analysis we refer the reader to the work of Joachims et al. (2009). We conclude this section with three important remarks on the training of SSVM. First, cutting-plane algorithm requires $\mathcal{O}(n)$ calculations of the function $f(x)$ which involves a maximization problem. In many applications where the output space has finite cardinality (*e.g.* multilabel classification) this is easily solvable, but when the output space is dense, there's often no immediate solution. Second, each iteration of the algorithm requires solving a quadratic programming problem, which is in general a NP-hard task, so extra care is required in the selection of such algorithm. Third, these formulations imply $|\mathcal{Y}|^n$ constraints, therefore they are intractable for dense output spaces such as constrained subsets of $\mathbb{R}^d$ or differentiable manifolds.

2.2 Consistent Regularized Structured Prediction

Ciliberto et al. (2017) have proposed an approach to structured problems based on the implicit structure that a loss functions inherits from the output domain. This family of algorithms can be shown to be statistically consistent and are suitable for dense output spaces while retaining a polynomial computational complexity at training time. To introduce this framework, we first need define an important property of loss functions.

Definition 2.2.1 (Structure Encoding Loss Function (SELF))**.** *Given a separable Hilbert space $\mathcal{F}_\mathcal{H}$ with inner product $\langle \cdot, \cdot \rangle_{\mathcal{F}_\mathcal{H}}$, a function $\Delta \colon \mathcal{Y} \times \mathcal{Y} \to \mathbb{R}$ is a **structure encoding loss** function iff there exists a continuous embedding $\psi \colon \mathcal{Y} \to \mathcal{F}_\mathcal{H}$ and a bounded linear operator $V \colon \mathcal{F}_\mathcal{Y} \to \mathcal{F}_\mathcal{H}$ such that*

$$\Delta(y, y') = \big\langle \psi(y), V\psi(y') \big\rangle_{\mathcal{F}_\mathcal{H}} \qquad \forall\, y, y' \in \mathcal{Y} \tag{2.9}$$

This property is indeed similar to the property of reproducing kernels introduced in Equation (1.19), however Δ does not require to be symmetric because the operator V is not, in general, symmetric. Intuitively, this suggests that whatever structure the output space $\mathcal{Y}$ has, it is possible to compare two of its element in some Hilbert space $\mathcal{F}_\mathcal{Y}$ by means of some map ψ and some operator V. Moreover, as observed in Ciliberto et al. (2016), a wide range of loss functions often used in machine learning are SELF. Examples include any loss on $\mathcal{Y}$ of finite cardinality, Hinge loss and Least-Squares loss. Having established the concept of SELF loss, we can now introduce a formulation of the consistent structured estimator.

Definition 2.2.2 (Consistent Regularized Structured Predictor (CRiSP))**.** *Given a SELF loss* $\Delta\colon \mathcal{Y}\times\mathcal{Y}\to\mathbb{R}$ *and a dataset* $\{x_i, y_i\}_{i=1}^n$ *sampled from a distribution* $\rho(x,y)$*, the following function is a consistent* ***estimator*** *for Equation* (1.1)

$$\widehat{f}(x) = \operatorname*{arg\,min}_{y\in\mathcal{Y}} \sum_{i=1}^{n} \alpha_i(x)\,\Delta(y, y_i) \tag{2.10}$$

with

$$\alpha(x) = [\alpha_1, \ldots, \alpha_n]^\top = \left(K + n\lambda I_n\right)^{-1} K_x \tag{2.11}$$

where K *is the kernel matrix of a fixed kernel* $k\colon \mathcal{X}\times\mathcal{X}\to\mathbb{R}$*, with entries* $\{K\}_{ij} = k(x_i, x_j)$*,* $K_x \in \mathbb{R}$ *is the vector with entries* $\{K_X\}_i = k(x, x_i)$ *and* I_n *is the identity matrix of size* n*.*

Conceptually, it is possible to divide training and inference in two steps. Training corresponds to computing the vector of weights α, while inference consists in finding the maximizing y for the computed α. The key differences with SSVM are two. The training complexity for computing Equation (2.11) is $\mathcal{O}(n^3)$. However there are no hidden computational costs in the procedure since it consists mainly in inverting a positive definite matrix. Second, there are no constraints in the minimization problem, therefore is it possible to work with dense output spaces. The CRiSP estimator is similar to Equation (2.1) since it requires to solve a minimization problem at each evaluation, but the encoding and decoding functions are not immediatel clear. To clarify on these, we discuss the derivation of Equation (2.10).

Derivation of CRiSP

We show briefly how the estimator in Equation (2.10) is obtained. We begin by rewriting the expected risk in Equation (3.6) as

$$\mathcal{E}(f) = \int_{\mathcal{X}} \left\langle \psi(f(x)), V \int_{\mathcal{Y}} \psi(y)\, d\rho(y|x) \right\rangle_{\mathcal{F}_\mathcal{H}} d\rho(x) \tag{2.12}$$

where we have conditioned $\rho(y,x) = \rho(y|x)\rho_x(x)$, and used the SELF property of the loss. Any function $f^*\colon \mathcal{X}\to\mathcal{Y}$ minimizing the above functional must satisfy the following condition

$$f^*(x) = \operatorname*{arg\,min}_{y\in\mathcal{Y}} \left\langle \psi(y), V g^*(x) \right\rangle_{\mathcal{F}_\mathcal{H}} \tag{2.13}$$

with

$$g^*(x) = \int_{\mathcal{Y}} \psi(y)\, d\rho(y|x) \tag{2.14}$$

where we have introduced the function $g^* : \mathcal{X} \to \mathcal{F}_{\mathcal{H}}$ that maps each point $x \in \mathcal{X}$ to the conditional expectation of $\psi(y)$ given x. Here $h_w(y,v) = \langle \psi(y), v \rangle_{\mathcal{F}_{\mathcal{H}}}$ is the auxiliary similarity function of Equation (2.1) but it does not depend on any learnable parameter, while $e_w(x) = g^*(x)$ plays the role of the encoding function. Notice how we cannot compute it explicitly because we do not have access to $\rho(y,x)$. But noting that it minimizes the expected least squares error

$$\int \|\psi(y) - g(x)\|^2_{\mathcal{F}_{\mathcal{H}}} d\rho(x,y) \tag{2.15}$$

suggests that a least squares estimator can be considered as a substitute. We first illustrate this idea for $\mathcal{X} = \mathbb{R}^d$ and $\mathcal{F}_{\mathcal{H}} = \mathbb{R}^k$, while noting that the procedure can easily be expanded with kernel functions. In this case we can consider a ridge regression estimator for g^*

$$\widehat{g}(x) = \widehat{W}^\top x \tag{2.16}$$

which can be computed in closed form as

$$\widehat{W} = \operatorname*{arg\,min}_{W \in \mathbb{R}^{d\times k}} \frac{1}{n} \|XW - \psi(Y)\|_F^2 + \lambda \|W\|_F^2 \tag{2.17}$$

where $X = (x_1, \ldots, x_n)^\top \in \mathbb{R}^{n\times d}$ and $\psi(Y) = (\psi(y_1), \ldots, \psi(y_n))^\top \in \mathbb{R}^{n\times k}$ are the matrices whose i-th row correspond respectively to the training sample $x_i \in \mathcal{X}$ and the (mapped) training output $\psi(y_i) \in \mathcal{F}_{\mathcal{H}}$. We have denoted $\|\cdot\|_F^2$ the squared Frobenius norm of a matrix, namely the sum of all its squared entries. The ridge regression solution can be obtained in closed form as $\widehat{W} = (X^\top X + n\lambda I)^{-1} X^\top \psi(Y)$. For any $x \in \mathcal{X}$ we have

$$\begin{aligned}
\widehat{g}(x) &= \psi(Y)^\top X (X^\top X + n\lambda I)^{-1} x \\
&= \psi(Y)^\top \alpha(x) \\
&= \sum_{i=1}^n \alpha_i(x) \psi(y_i)
\end{aligned}$$

where we have introduced the coefficients $\alpha(x) = X(X^\top X + n\lambda I)^{-1}x \in \mathbb{R}^n$. By substituting $\widehat{g}$ to g^* in Equation (2.13) we have

$$\widehat{f}(x) = \operatorname*{argmin}_{y\in\mathcal{M}} \left\langle \psi(y), V\left(\sum_{i=1}^n \alpha_i(x)\psi(y_i)\right)\right\rangle = \operatorname*{argmin}_{y\in\mathcal{M}} \sum_{i=1}^n \alpha_i(x)\Delta(y,y_i) \tag{2.18}$$

where we have used the linearity of the sum and the inner product to move the coefficients α_i outside of the inner product. Since the loss is SELF, we then obtain $\langle \psi(y), V\psi(y_i)\rangle = \Delta(y,y_i)$ for any y_i in the training set. This recovers the estimator $\widehat{f}$ introduced in Equation (3.8), as desired.

Given a positive definite kernel the above idea can be extended. We can consider $\mathcal{X}$ to be a set and $k : \mathcal{X}\times\mathcal{X} \to \mathbb{R}$ a positive definite kernel. Then $\widehat{g}$ can be computed by kernel ridge regression (see Chapter 1) to obtain the scores $\alpha(x) = (K + n\lambda I)^{-1}K_x$. We end noting that the above discussion applies if $\mathcal{F}_\mathcal{H}$ is infinite dimensional. Indeed, thanks to the SELF assumption and the fact that we chose a ridge regression estimator, $\widehat{f}$ does not depend on explicit knowledge of the space $\mathcal{F}_\mathcal{H}$ but only on the loss function.

2.3 Neural Network Models

Artificial neural networks (NN) are a family of parametric predictive estimators that has recently shown remarkable results in various machine learning applications, particularly computer vision and natural language processing. The origin of these models is dated 1943 when McCulloch and Pitts (1943) postulated a first mathematical model of neural interactions in the human brain. This first effort paved the way to the perceptron (Werbos, 1974), an algorithm for binary classification, inspired by the biophysical model of a single neuron. While more developements took place in the following 40 years (Rumelhart et al., 1988; Weng et al., 1993), it is not until more recent times that these models have obtained outstanding empirical results (Krizhevsky et al., 2012), stirring the attention of the machine learning community. This results were made possible mostly thanks to two factors. The introduction of a compositional structure in NN model design called *deep learning* (Schmidhuber, 1992) and the accessibility of modern graphical processing units (GPU) that made possible to exploit the inherently parallel structure of these models to achieve faster training times. NN usually involve a careful design process to define a model fit to the task at hand, especially for structured datasets. We start by introducing a general form of encoding functions used in NN. We will illustrate an example of decoding function after commenting on the encoding.

Definition 2.3.1 (Neural network encoding function)**.** *A neural network **encoding** function is a composition of functions $f_l\colon \mathbb{R}^{u_{l-1}} \to \mathbb{R}^{u_l}$ called layers, layers are defined as*

$$f_l(x) = \sigma_l(W_l x) \tag{2.19}$$

where $W \in \mathbb{R}^{u_l \times u_{l-1}}$ is a matrix of learnable parameters and $\sigma_l\colon \mathbb{R}^{u_l} \to \mathbb{R}^{u_l}$ is a non linear function applied element-wise to a vector. The number u_l is often referred to as the number of hidden units of the l-th layer.
Given a number of layers L, the encoding function of a neural network is defined as

$$e_w(x) = (f_L \circ f_{L-1} \circ \ldots \circ f_2 \circ f_1)(x) \tag{2.20}$$

Some common non-linearities are the positive part (ReLU (Hahnloser et al., 2000)), the sigmoid function and the hyperbolic tangent (LeCun et al., 2012). While farily general, this definition already captures the fundamental ideas in NN. At its core NN are the combination of a linear operator and a non-linear operation, however these basic building blocks are used to create composite structures, built by "stacking" layers one after the other. This way of building models is called *deep learning* and it is hypothesized that it is one reason of the empirical success of NN (Goodfellow et al., 2016). Another important feature is the possibility of imposing different kind of structures on the weights $\{W_l\}_{l=1}^{L}$ to leverage prior knowledge on the problem. One of the most succesful structures is the convolutional neural network (Fukushima, 1979), which effectively transforms Wx into a product with a sparser matrix where weights are shared between different rows of W. This results in faster training and other desirable properties for computer vision tasks. When no structure is imposed, the layers are called *fully connected* layers. It's important to notice that when trying to minimize the empirical risk using a NN the resulting problem is non-convex. Common practice is to approach the minimization problem with iterative gradient based techniques, of which the most common are gradient descent (Cauchy, 1847) or related variants such as Adam optimizer (Kingma and Ba, 2014). On one hand because the problem is non-convex, there are no guarantess that the found solution is the global minimum, on the other hand, all this variants can be implemented in their stochastic counterparts which allow to perform optimization on large scale dataset with good empirical results (Bottou and Bousquet, 2008).
We discussed some ideas behind that encoding function, however we did not touch yet on the decoding function used by NNs. Because designing a NN model is highly dependent on the target task, decoding functions tend to be defined ad-hoc. To illustrate such

procedure, in the following we go over two examples. The first introduces the reader to one of the most common types of decoding function for classification tasks. The second is an alternative example designed specifically for sequence-to-sequence tasks.

2.3.1 Segmentation with Neural Networks

The goal of segmentation consists in labelling each pixel of an image with one of L possible classes. Therefore the task is to find a map $\widehat{f}\colon \mathbb{R}^{PC} \to \{1,\dots,L\}^P$ where P is the number of pixel per image and C the number of color channels. We denote with $x \in \mathbb{R}^{PC}$ the vectorized image and with $y^{(j)}$ the i-th element of the sequence of labels y. A naive approach would train P classifiers independently, one for each pixel of the image. In contrast to this we introduce a more recent deep learning based approach for segmentation, proposed by Chen et al. (2018). The model uses an encoder function in the form of Equation (2.20), where the learnable weights are defined to be convolutional operators. The decoder function is defined as

$$d_w(y, e_w(x)) = \underset{y\in\{1,\dots,L\}^P}{\arg\max} \sum_{i=1}^{P} h_w^{(i)}(y^{(i)}, e_w(x)) \tag{2.21}$$

with $h_w^{(i)}(y^{(i)}, x) = (f_{Mj} \circ \dots \circ f_{1j})(x)$ where $f_{mj} \forall\ m \in \{1,\dots,M\}$ are convolutional layers. In practice $h_w^{(i)}\colon \mathcal{Y} \times \mathcal{F}_{\mathcal{H}} \to \mathbb{R}$ computes the likelihood that, given the input x, its i-th pixel belongs to one of the L classes. Intuitively, $h_w \circ e_w$ approximates an unnormalized probability distribution over the L classes for each pixel, and selects the most likely class. Having one $h^{(i)}$ for pixel decouples the decoding process across different pixels, hiding the inter-pixel dependence. This results in an estimator with fast decoding and good accuracy when compared to the naive approach.

2.3.2 Recurrent Neural networks

Another common task is sequence to sequence classification. The goals is to label a sequence of samples $s_P = \{s_t\}_{t=1}^P$, $s_t \in \mathbb{R}^d$, with a sequence of labels $l \in \{1,\dots,L\}^P$. Similarly to segmentation tasks, the output space grows exponentially in the number of labels and the problem becomes quickly too computational expensive for longer sequences. Hochreiter and Schmidhuber (1997) have proposed Recurrent Neural Networks (RNN) as a model specifically designed to deal with this task in a more efficient way.

Definition 2.3.2 (Recurrent Neural Network)**.** *Given a sequence of vectors s_P, a recurrent neural network **encoder** function is defined iteratively as*

$$e_w(s_t) = \sigma_1\big(W_1 m_t + b_1\big) \tag{2.22}$$

where

$$m_t = \sigma_2\big(W_2 s_t + U e_w(s_{t-1}) + b_2\big) \tag{2.23}$$

*and $W_1 \in \mathbb{R}^{h\times d}$, $b_1 \in \mathbb{R}^h$, $U \in \mathbb{R}^{h\times k}$ $W_2 \in \mathbb{R}^{k\times h}$, $b_2 \in \mathbb{R}^k$ are parameters to be learned. σ_1, σ_2 are two element-wise non-linear functions. The **decoding** function is defined as in Equation (2.21) over the elements of the sequence instead of pixels.*

Intuitively, RNN keeps an internal hidden state (the vector m_t) that encodes the dependencies of current sample with respect to previous samples. Notice that for recurrent neural networks the decoding function stays the same, however the sample-by-sample structure of the encoding function allows to decode sequence elements one at a time. This allows for faster inference times. Moreover, this model is agnostic to the length of the sequence and therefore can be used for sequences of varying length without the need for any modification.

We make a final note on the loss functions for structured NN. Both of the models we introduced are usually trained using a cross-entropy loss after trasforming the output values of $h_w^{(i)}$ into a distribution over the labels with the softmax function $s\colon \mathbb{R} \to [0,1]$. Specifically, the softmax function computed for one pixel or one sequence sample is

$$s(z_i) = \frac{e^{x_i}}{\sum_{j=1}^{L} e^{x_j}} \tag{2.24}$$

where $z_i = h_w^{(i)}\big(y^{(i)}, e_w(x)\big)$. And the cross-entropy loss for one sample is

$$\min_{h_w, e_w} -\sum_{i=1}^{P}\sum_{j=1}^{L} \mathbb{1}_{y^{(i)}=j} \log\Big(s\big(z_1\big)\Big) \tag{2.25}$$

where the dependency of the problem with respect to w is hidden in z_i. In a sense, this approach recovers the idea of probabilistic graphical models, where, for structured output, it is a good choice to estimate a score proportional to the maximum likelihood estimator over the possible outputs.

Part II

Manifold Structured Machine Learning

Chapter 3

Manifold Structured Prediction

3.1 Introduction

In Chapter 1 we talked about regression and classification as the most common machine learning applications, however in Chapter 2 we introduced how it is often interesting to estimate functions with more structured outputs. When the output space can be assumed to be a vector space, many ideas from regression can be extended, think for example to multivariate (Härdle and Simar, 2007) or functional regression (Morris, 2015). However, a lack of a natural vector structure is a feature of many practically interesting problems, such as ranking (Duchi et al., 2010), quantile estimation (Takeuchi et al., 2006) or graph prediction (Paaßen et al., 2017). In this latter case, the outputs are typically provided only with some distance or similarity function that can be used to design appropriate loss function. Knowledge of the loss is sufficient to analyze an abstract empirical risk minimization approach within the framework of statistical learning theory, but deriving approaches that are at the same time statistically sound and computationally feasible is a key challenge. While ad-hoc solutions are available for many specific problems (Bicer et al., 2011; Daume and Marcu, 2006; Kadous and Sammut, 2005; Nowozin et al., 2011), structured prediction (Bakir et al., 2007) provides a unifying framework where a variety of problems can be tackled as special cases.

Classically, structured prediction considers problems with finite, albeit potentially huge, output spaces. In this chapter, we study how these ideas can be applied to dense output spaces. In particular, we consider the case where the output space is a Riemannian manifold, that is the problem of manifold structured prediction (also called manifold-valued regression (Steinke and Hein, 2009)). While also in this case ad-hoc methods are available (Steinke et al., 2010), in this chapter we adopt and study a structured prediction

approach starting from a framework proposed in (Ciliberto et al., 2016). Within this framework, it is possible to derive a statistically sound, and yet computationally feasible, structured prediction approach, as long as the loss function satisfies a suitable structural assumption. Moreover we can guarantee that the computed prediction is always an element of the manifold.
The rest of the chapter is organized as follows. We introduce some basic differential geometry concepts in Section 3.2. In Section 3.3, we define the problem and explain the proposed algorithm. In Section 3.4 we state and prove the theoretical results of this work. In Section 3.5 we explain how to compute the proposed algorithm and we show the performance of our method on synthetic and real data.

3.2 Differential geometry

We introduce briefly the fundamental objects of differential geometry, for a more in-depth explanation, we refer the reader to the excellent *Optimization Algorithms on Matrix Manifolds* (Absil et al., 2009).
Intuitively, manifold are topological sets that do not support a vector structure but can be mapped everywhere to Euclidean space by means of a continuous invertible function. By exploiting this mapping it is possible to induce local vector spaces, this allows for operations such as differentiating functions or computing distances.
The simplest example of a manifold is the sphere. Consider the sphere as a surface in a 3-dimensional Euclidean space, it is easy to notice that adding two vectors $x_1, x_2 \in \mathcal{S}_2 = \{x \in \mathbb{R}^3 : \|x\|_2 = 1\}$, the resulting vector is not a point of the sphere. However it is possible to find functions that map open sets of the sphere to $\mathbb{R}^2$, where usual linear algebra operations are well defined.
More formally, the definition of topological manifold is:

Definition 3.2.1 (Topological manifold)**.** *A **topological manifold** $\mathcal{M}$ of dimension d is a topological space $\mathcal{M}$, such that every point $x \in \mathcal{M}$ has a neighbourhood $U \subset \mathcal{M}$ which is homeomorphic to an open set $\phi(U)$ in Euclidean space $\mathbb{R}^d$. The homeomorphism $\phi\colon U \to \mathbb{R}^d$ is called **chart** on U and is denoted by (U, ϕ).*

Charts introduce a tool to define local coordinates. To guarantee that it is possible to move from one set of coordinates to another we first introduce the transition function:

Definition 3.2.2 (Compatible charts)**.** *Given two coordinate charts* (ϕ, U), (ψ, V), *the* ***transition function*** *is defined as:*

$$\phi \circ \psi^{-1} \colon \psi(U \cup V) \subset \mathbb{R}^d \to \phi(U \cup V) \subset \mathbb{R}^d \tag{3.1}$$

everywhere where $U \cap V \neq \emptyset$. *Two charts are* ***compatible*** *if the transition maps are smooth.*

If a manifold $\mathcal{M}$ has only pair-wise compatible charts, the manifold is **differentiable**. Indeed, composing a real-valued function $f \colon \mathcal{M} \to \mathbb{R}$ with a chart inverse $\phi \colon \mathbb{R}^d \to \mathcal{M}$ allows us to differentiate the function. Moreover, chart compatibility guarantees that this operation is continuous even when changing charts.
It is now possible to define tangent vectors.

Definition 3.2.3 (Tangent vector)**.** *Let* $\gamma \colon [0,1] \to M$ *be a smooth map such that* $\gamma(0) = p$. *Let* $\mathcal{F}_p(\mathcal{M})$ *denote the set of smooth real-valued functions defined on a neighbourhood of* p. *Then a* ***tangent vector*** ξ_p *to a manifold* $\mathcal{M}$ *at a point* p *is an operator from* $\mathcal{F}_p(\mathcal{M})$ *to* $\mathbb{R}$ *such that there exists a curve* $\gamma \colon [0,1] \to \mathcal{M}$ *with* $\gamma(0) = 1$ *satisfying:*

$$\xi_p f = \gamma'(0) f = \left. \frac{d(f(\gamma(t))}{dt} \right|_{t=0} \tag{3.2}$$

The set of all tangent vectors at p *is called* ***tangent space****, denoted by* $\mathcal{T}_p\mathcal{M}$.

Intuitively, by introducing trajectories γ on manifolds, given a fixed function f, we have established a local equivalence between the derivative of such trajectory and elements of the set $\mathcal{T}_p\mathcal{M}$. It is then easy to show that the elements of $\mathcal{T}_p\mathcal{M}$ have a vector structure and satisfy Lebniz's rule of derivation.

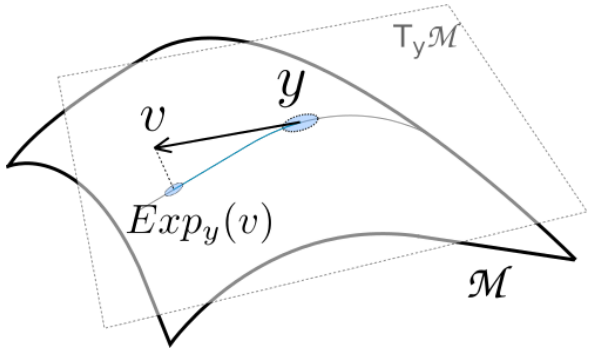


Figure 3.1 A representation of a tangent space and the exponential map mapping the vector v to the corresponding element in in the manifold.

3.2.1 Riemannian manifold

Having defined tangent vectors, it is possible to impose further structure on the manifold by introducing Riemannian metrics. Riemannian metrics are a generalization of inner products in the tangent bundle.

Definition 3.2.4 (Riemannian metric)**.** *A **Riemannian metric** g is a family of positive-definite inner products $g_p \colon \mathcal{T}_p\mathcal{M} \times \mathcal{T}_p\mathcal{M} \to \mathbb{R}$, one for each tangent space of $\mathcal{M}$. This family is smooth in the sense that for any two smooth vector fields X and Y, the function $p \in \mathcal{M} \to g_p(X_p, Y_p)$ is smooth.*

The Riemannian metric allows us to define length of curves on the manifold.

Definition 3.2.5 (Curve length)**.** *Given a smooth curve $\gamma \colon [0,1] \to \mathcal{M}$, its **length** is defined as:*

$$L(\gamma) = \int_0^1 \sqrt{g_{\gamma(t)}\big(\gamma'(t), \gamma'(t)\big)}\, dt \tag{3.3}$$

Where we abused the notation by using $\gamma'(t)$ to denote the set of corresponding tangent vectors along $\gamma(t)$ and $g_{\gamma(t)}$ to denote the Riemannian metric computed along the trajectory of points defined by $\gamma(t)$.
With this definition, it is then possible to define the distance between two elements of a manifold as the length of the shortest path between those two elements, the shortest path is called *geodesic curve* while the its length is the **geodesic distance**.
We finally introduce one last item that will be of use in the following: the exponential map.

Definition 3.2.6 (Exponential map)**.** *Given a vector $\xi_p \in \mathcal{T}_p\mathcal{M}$ such that there is a geodesic $\gamma \colon [0,1] \to \mathcal{M}$ satisfying*

$$\gamma(0) = p, \quad \frac{d\gamma}{dt}(0) = \xi_p \tag{3.4}$$

*then the **exponential map** $exp_p \colon \mathcal{T}_p\mathcal{M} \to \mathcal{M}$ of ξ_p is defined as:*

$$exp_p(\xi_p) = \gamma(1) \tag{3.5}$$

Intuitively, the exponential map maps a vector from the tangent space $\mathcal{T}_p\mathcal{M}$ to the point at the end of the geodesic curve having that vector as tangent vector at its start (see Figure 3.1).
The exponential map is a local diffeomorphism, the minimum geodesic distance for which every exponential map is invertible on the manifold is called **injectivity radius**, we will denote it with $\rho_{\mathcal{M}}$.

Interestingly, the notion of convexity can be generalized to geodesics. A subset U of a Riemannian manifold is **geodesically convex** if there is a one and only one geodesic curve $\gamma\colon [0,1] \to U$ between any two points in U.

3.3 Structured Prediction for Manifold Valued Regression

As we have discussed, the goal of supervised learning is to find a functional relation between an input space $\mathcal{X}$ and an output space $\mathcal{Y}$ given a finite set of observations. Traditionally, the output space is either a linear space (e.g. $\mathcal{Y} = \mathbb{R}^M$) or a discrete set (e.g. $\mathcal{Y} = \{0,1\}$ in binary classification). We will consider the problem of learning manifold-valued functions (Steinke et al., 2010), in which output data lies on a manifold $\mathcal{M} \subset \mathbb{R}^d$. In this context, statistical learning corresponds to solving

$$\underset{f \in \mathcal{X} \to \mathcal{Y}}{\arg\min}\, \mathcal{E}(f) \tag{3.6}$$

with

$$\mathcal{E}(f) = \int_{\mathcal{X}\times\mathcal{Y}} \Delta(f(x), y)\, \rho(x,y) \tag{3.7}$$

where $\mathcal{Y}$ is a subset of the manifold $\mathcal{M}$ and ρ is an unknown distribution on $\mathcal{X} \times \mathcal{Y}$. Here, $\Delta\colon \mathcal{Y} \times \mathcal{Y} \to \mathbb{R}$ is a loss function that measures prediction errors for points estimated on the manifold. The minimization is meant over the set of all measurable functions from $\mathcal{X}$ to $\mathcal{Y}$. The distribution is fixed but unknown and a learning algorithm seeks an estimator $\widehat{f}\colon \mathcal{X} \to \mathcal{Y}$ that approximately solves Equation (3.6), given a set of training points $(x_i, y_i)_{i=1}^n$ sampled independently from ρ.

A concrete example of loss function that we will consider in this work is $\Delta = d^2$ the squared geodesic distance $d : \mathcal{Y} \times \mathcal{Y} \to \mathbb{R}$ (Lee, 2003). As we have seen, the geodesic distance is the natural generalization to distances on a Riemannian manifold and is a natural loss function in the context of manifold regression (Fletcher, 2013; Hauberg et al., 2012; Hinkle et al., 2012; Steinke and Hein, 2009; Steinke et al., 2010).

3.3.1 Manifold Valued Regression via Structured Prediction

To tackle the problem of manifold structured prediction we will consider the CRiSP approach to manifold valued regression following ideas introduced in Section 2.2, which we recall here. Given a training set $\{x_i, y_i\}_{i=1}^n$, an estimator for problem Equation (3.6) is

defined by

$$\widehat{f}(x) = \operatorname*{arg\,min}_{y \in \mathcal{Y}} \sum_{i=1}^{n} \alpha_i(x) \Delta(y, y_i) \tag{3.8}$$

for any $x \in \mathcal{X}$. The coefficients $\alpha(x) = (\alpha_1(x), \dots, \alpha_n(x))^\top \in \mathbb{R}^n$ are obtained solving a linear system for a problem akin to kernel ridge regression (see Section 2.2): given a positive definite kernel $k : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$ over $\mathcal{X}$, we have

$$\alpha(x) = (\alpha_1(x), \dots, \alpha_n(x))^\top = (K + n\lambda I_n)^{-1} K_x \tag{3.9}$$

where $K \in \mathbb{R}^{n \times n}$ is the empirical kernel matrix with $K_{i,j} = k(x_i, x_j)$, and $K_x \in \mathbb{R}^n$ the vector whose i-th entry corresponds to $(K_x)_i = k(x, x_i)$.
Computing the estimator in Equation (3.8) involves two steps. During a *training step* the score function $\alpha : \mathcal{X} \to \mathbb{R}^n$ is learned, while during the *prediction step*, the output $\widehat{f}(x) \in \mathcal{Y}$ is estimated on a new test point $x \in \mathcal{X}$. This last step requires minimizing the linear combination of distances $\Delta(y, y_i)$ between a candidate $y \in \mathcal{Y}$ and the training outputs $(y_i)_{i=1}^n$, weighted by the corresponding scores $\alpha_i(x)$.
We next discuss the main theoretical result of this chapter, showing that a large class of loss functions for manifold structured prediction are SELF. This will allow us to prove consistency and learning rates for the manifold structured estimator considered in this work.

3.4 Characterization of SELF Function on Manifolds

In this section we provide sufficient conditions for a wide class of functions on manifolds to satisfy the definition of SELF. A key example will be the case of the squared geodesic distance. To this end we will make the following assumptions on the manifold $\mathcal{M}$ and the output space $\mathcal{Y} \subseteq \mathcal{M}$ where the learning problem takes place.

Assumption 3.4.1. *$\mathcal{M}$ is a complete d-dimensional smooth connected Riemannian manifold, without boundary, with Ricci curvature bounded below and positive injectivity radius.*

The assumption above imposes basic regularity conditions on the output manifold. In particular we require the manifold to be locally diffeomorphic to $\mathbb{R}^d$ and that the tangent space of $\mathcal{M}$ at any $p \in \mathcal{M}$ varies smoothly with respect to p. This assumption avoids pathological manifolds and is satisfied for instance by any smooth compact manifold (e.g. the sphere, torus, etc.) (Lee, 2003). Other notable examples are the statistical manifold

(without boundary) (Amari and Nagaoka, 2007) any open bounded sub-manifold of the cone of positive definite matrices, which is often studied in geometric optimization settings (Absil et al., 2009). This assumption will be instrumental to guarantee the existence of a space of functions $\mathcal{F}_\mathcal{H}$ on $\mathcal{M}$ rich enough to contain the squared geodesic distance.

Assumption 3.4.2. *$\mathcal{Y}$ is a compact geodesically convex subset of the manifold $\mathcal{M}$.*

The effect of Assumption 3.4.2 is twofold. On one hand it guarantees a generalized notion of convexity for the space $\mathcal{Y}$ on which we will solve the optimization problem in Equation (3.8). On the other hand it avoids the geodesic distance to have singularities on $\mathcal{Y}$ (which is key to our main result below). We are ready to prove the main result of this chapter.

Theorem 3.4.1 (Smooth Functions are SELF)**.** *Let $\mathcal{M}$ satisfy Assumption 3.4.1 and $\mathcal{Y} \subseteq \mathcal{M}$ satisfy Assumption 3.4.2. Then, any smooth function $h : \mathcal{Y} \times \mathcal{Y} \to \mathbb{R}$ is SELF on $\mathcal{Y}$.*

Sketch of proof of Theorem 3.4.1. The complete proof of Theorem 3.4.1 is reported in Section A.1. The proof hinges around the following key steps:

Step 1 If there exists an RKHS $\mathcal{F}_\mathcal{H}$ on $\mathcal{M}$, then any $h \in \mathcal{F}_\mathcal{H} \otimes \mathcal{F}_\mathcal{H}$ is SELF. Let $\mathcal{F}_\mathcal{H}$ be a reproducing kernel Hilbert space (RKHS) (Aronszajn, 1950) of functions on $\mathcal{M}$ with associated bounded kernel $k : \mathcal{M} \times \mathcal{M} \to \mathbb{R}$. Let $\mathcal{F}_\mathcal{H} \otimes \mathcal{F}_\mathcal{H}$ denote the RKHS of functions $h : \mathcal{M} \times \mathcal{M} \to \mathbb{R}$ with associated kernel $\bar{k}$ such that $\bar{k}((y,z),(y',z')) = k(y,y')k(z,z')$ for any $y, y', z, z' \in \mathcal{M}$. Let, $h : \mathcal{M} \times \mathcal{M} \to \mathbb{R}$ be such that $h \in \mathcal{F}_\mathcal{H} \otimes \mathcal{F}_\mathcal{H}$. Recall that $\mathcal{F}_\mathcal{H} \otimes \mathcal{F}_\mathcal{H}$ is isometric to the space of Hilbert-Schmidt operators from $\mathcal{F}_\mathcal{H}$ to itself. Let $V_h : \mathcal{F}_\mathcal{H} \to \mathcal{F}_\mathcal{H}$ be the operator corresponding to h via such isometry. We show that the SELF definition is satisfied with $V = V_h$ and $\psi(y) = k(y,\cdot) \in \mathcal{F}_\mathcal{H}$ for any $y \in \mathcal{M}$. In particular, we have $\|V\| \leq \|V\|_{\mathrm{HS}} = \|h\|_{\mathcal{F}_\mathcal{H} \otimes \mathcal{F}_\mathcal{H}}$, with $\|V\|_{\mathrm{HS}}$ denoting the Hilbert-Schmidt norm of V.

Step 2: Under Assumption 3.4.2, $C_c^\infty(\mathcal{M}) \otimes C_c^\infty(\mathcal{M})$ "contains" $C^\infty(\mathcal{Y} \times \mathcal{Y})$. If $\mathcal{Y}$ is compact and geodesically convex, then it is diffeomorphic to a compact set of $\mathbb{R}^d$. By using this fact, we prove that any function in $C^\infty(\mathcal{Y} \times \mathcal{Y})$, the space of smooth functions on $\mathcal{Y} \times \mathcal{Y}$, admits an extension in $C_c^\infty(\mathcal{M} \times \mathcal{M})$ the space of smooth functions on $\mathcal{M} \times \mathcal{M}$ vanishing at infinity (this is well defined since $\mathcal{M}$ is diffeomorphic to $\mathbb{R}^d$ thanks to Assumption 3.4.1), and that $C_c^\infty(\mathcal{M} \times \mathcal{M}) = C_c^\infty(\mathcal{M}) \otimes C_c^\infty(\mathcal{M})$.

Step 3: Under Assumption 3.4.1, there exists an RKHS on $\mathcal{M}$ containing $C_c^\infty(\mathcal{M})$. Under Assumption 3.4.1, the Sobolev space $\mathcal{F}_\mathcal{H} = H_s^2(\mathcal{M})$ of square integrable functions with

smoothness s is an RKHS for any $s > d/2$ (see (Hebey, 2000) for a definition of Sobolev spaces on Riemannian manifolds).

The proof proceeds as follows: from Step 1, we see that to guarantee h to be SELF it is sufficient to prove the existence of an RKHS $\mathcal{F}_\mathcal{H}$ such that $h \in \mathcal{F}_\mathcal{H} \otimes \mathcal{F}_\mathcal{H}$. The rest of the proof is therefore devoted to showing that for smooth functions this is satisfied for $\mathcal{F}_\mathcal{H} = H^2_s(\mathcal{M})$. Since h is smooth, by Step 2 we have that under Assumption 3.4.2, there exists a $\bar{h} \in C^\infty_c(\mathcal{M}) \otimes C^\infty_c(\mathcal{M})$ whose restriction $\bar{h}|_{\mathcal{Y}\times\mathcal{Y}}$ to $\mathcal{Y} \times \mathcal{Y}$ corresponds to h. Now, denote by $H^2_s(\mathcal{M})$ the Sobolev space of squared integrable functions on $\mathcal{M}$ with smoothness index $s > 0$. By construction (Hebey, 2000), for any $s > 0$, we have $C^\infty_c(\mathcal{M})|_\mathcal{Y} \subseteq H^2_s(\mathcal{M})|_\mathcal{Y}$, namely for any function. In particular, $\bar{h} \in C^\infty_c(\mathcal{M}) \otimes C^\infty_c(\mathcal{M}) \subseteq H^2_s(\mathcal{M}) \otimes H^2_s(\mathcal{M})$. Finally, Step 3 guarantees that under Assumption 3.4.1, $\mathcal{F}_\mathcal{H} = H^2_s(\mathcal{M})$ with $s > d/2$ is an RKHS, showing that $h \in \mathcal{F}_\mathcal{H} \otimes \mathcal{F}_\mathcal{H}$ as desired. □

Interestingly, Theorem 3.4.1 shows that the SELF estimator proposed in Equation (3.8) can tackle *any* manifold valued learning problem in the form of Equation (3.6) with smooth loss function. In the following we study the specific case of the squared geodesic distance.

Theorem 3.4.2 (d^2 is SELF)**.** *Let $\mathcal{M}$ satisfy Assumption 3.4.1 and $\mathcal{Y} \subseteq \mathcal{M}$ satisfy Assumption 3.4.2. Then, the squared geodesic distance $\Delta = d^2 : \mathcal{M} \times \mathcal{M} \to \mathbb{R}$ is smooth on $\mathcal{Y}$. Therefore Δ is SELF on $\mathcal{Y}$.*

The proof of the result above is reported in the supplementary material. The main technical aspect is to show that regularity provided by Assumption 3.4.2 guarantees the squared geodesic distance to be smooth. The fact that Δ is SELF is then an immediate corollary of Theorem 3.4.1.

3.4.1 Statistical Properties of Manifold Structured Prediction

In this section, we characterize the generalization properties of the manifold structured estimator Equation (3.8) in light of Theorem 3.4.1 and Theorem 3.4.2.

Theorem 3.4.3 (Universal Consistency)**.** *Let $\mathcal{M}$ satisfy Assumption 3.4.1 and $\mathcal{Y} \subseteq \mathcal{M}$ satisfy Assumption 3.4.2. Let $\mathcal{X}$ be a compact set and $k : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$ be a bounded continuous universal kernel*[1] *For any $n \in \mathbb{N}$ and any distribution ρ on $\mathcal{X} \times \mathcal{Y}$ let $\widehat{f}_n : \mathcal{X} \to \mathcal{Y}$ be the manifold structured estimator in Equation (3.8) for a learning problem with smooth loss*

[1] This is standard assumption for universal consistency (see (Steinwart and Christmann, 2008)). An example of continuous universal kernel on $\mathcal{X} = \mathbb{R}^d$ is the Gaussian $k(x, x') = \exp(-\|x - x'\|^2/\sigma)$, for $\sigma > 0$.

function Δ, *with* $(x_i, y_i)_{i=1}^n$ *training points independently sampled from* ρ *and* $\lambda_n = n^{-1/4}$. *Then*

$$\lim_{n\to\infty} \mathcal{E}(\widehat{f}_n) = \mathcal{E}(f^*) \quad \textit{with probability } 1.$$

The result above follows from Thm. 4 in A Consistent Regularization Approach for Structured Prediction (Ciliberto et al., 2016) combined with our result in Theorem 3.4.1. It guarantees that the algorithm considered in this work finds a consistent estimator for the manifold structured problem, when the loss function is smooth (thus also in the case of the squared geodesic distance). As it is standard in statistical learning theory, we can impose regularity conditions on the learning problem, in order to derive also generalization bounds for $\widehat{f}$. In particular, if we denote by $\mathcal{F}$ the RKHS associated to the kernel k, we will require g^* to belong to the same space $\mathcal{F}_\mathcal{H} \otimes \mathcal{F}$ where the estimator $\widehat{g}$ introduced in Equation (2.18) is learned. In the simplified case discussed in Section 2.2, with linear kernel on $\mathcal{X} = \mathbb{R}^d$ and $\mathcal{F}_\mathcal{H} = \mathbb{R}^k$ finite dimensional, we have $\mathcal{F} = \mathbb{R}^d$ and this assumption corresponds to require the existence of a matrix $W_*^\top \in \mathbb{R}^{k\times d} = \mathcal{F}_\mathcal{H} \otimes \mathcal{F}$, such that $g^*(x) = W_*^\top x$ for any $x \in \mathcal{X}$. In the general case, the space $\mathcal{F}_\mathcal{H} \otimes \mathcal{F}$ extends to the notion of *reproducing kernel Hilbert space for vector-valued functions* (see the works of Alvarez et al. (2012); Micchelli and Pontil (2005)) but the same intuition applies.

Theorem 3.4.4 (Generalization Bounds)**.** *Let* $\mathcal{M}$ *satisfy Assumption 3.4.1 and* $\mathcal{Y} \subseteq \mathcal{M}$ *satisfy Assumption 3.4.2. Let* $\mathcal{F}_\mathcal{H} = H_s^2(\mathcal{M})$ *with* $s > d/2$ *and* $k : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$ *be a bounded continuous reproducing kernel with associated RKHS* $\mathcal{F}$. *For any* $n \in \mathbb{N}$, *let* $\widehat{f}_n$ *denote the manifold structured estimator in Equation (3.8) for a learning problem with smooth loss* $\Delta : \mathcal{Y} \times \mathcal{Y} \to \mathbb{R}$ *and* $\lambda_n = n^{-1/2}$. *If the conditional mean* g^* *belongs to* $\mathcal{F}_\mathcal{H} \otimes \mathcal{F}$, *then*

$$\mathcal{E}(\widehat{f}_n) - \mathcal{E}(f^*) \leq \mathsf{c}_\Delta \mathsf{q}\, \tau^2\, n^{-\frac{1}{4}} \tag{3.10}$$

holds with probability not less than $1 - 8e^{-\tau}$ *for any* $\tau > 0$, *with* $\mathsf{c}_\Delta = \|\Delta\|_{\mathcal{F}_\mathcal{H}\otimes\mathcal{F}_\mathcal{H}}$ *and* q *a constant not depending on* n, τ *or the loss* Δ.

The generalization bound of Theorem 3.4.4 is obtained by adapting Thm. 5 of A Consistent Regularization Approach for Structured Prediction (Ciliberto et al., 2016) to our results in Theorem 3.4.1 as detailed in the supplementary material. To our knowledge these are the first results characterizing in such generality the generalization properties of an estimator for manifold structured learning with generic smooth loss function. We conclude with a remark on a key quantity in the bound of Theorem 3.4.4.

Remark 3.4.1 (The constant c_Δ)**.** *We comment on the role played in the learning rate by c_Δ, the norm of the loss function Δ seen an element of the Hilbert space $\mathcal{F}_\mathcal{H} \otimes \mathcal{F}_\mathcal{H}$. Indeed, from the discussion of Theorem 3.4.1 we have seen that any smooth function on $\mathcal{Y}$ is SELF and belongs to the set $\mathcal{F}_\mathcal{H} \otimes \mathcal{F}_\mathcal{H}$ with $\mathcal{F}_\mathcal{H} = H^2_s(\mathcal{M})$, the Sobolev space of squared integrable functions for $s > d/2$. Following this interpretation, we see that the bound in Theorem 3.4.4 can improve significantly (in terms of the constants) depending on the regularity of the loss function: smoother loss functions will result in "simpler" learning problems and vice-versa. In particular, when Δ corresponds to the squared geodesic distance, the more "regular" is the manifold $\mathcal{M}$, the learning problem will be. A refined quantitative characterization of c_Δ in terms of the Ricci curvature and the injective radius of the manifold is left to future work.*

3.5 Algorithm and Experimental applications

In this section we recall geometric optimization algorithms that can be adopted to perform the estimation of $\widehat{f}$ on a novel test point x. We then evaluate the performance of the proposed method in practice, reporting numerical results on simulated and real data.

3.5.1 Optimization on Manifolds

We begin discussing the computational aspects related to evaluating the manifold structured estimator. In particular, we discuss how to address the optimization problem in Equation (3.8) in specific settings. Optimization on Riemannian manifolds is a topic that has recently seen growing interest from the community given its application in numerical analysis and matrix factorization (Gower et al., 2004; Jae Hwang et al., 2015). The goal of optimization on manifolds is to solve

$$\min_{y \in \mathcal{Y}} F(y) \tag{3.11}$$

where in our case $F(y)$ corresponds to a linear combination of $\Delta(y, y_i)$ weighted by the scores $\alpha_i(x)$ computed according to Equation (3.9).
In the case where $\mathcal{Y}$ is a linear manifold or a subset of $\mathcal{M} = \mathbb{R}^d$, this problem is well studied (Peypouquet, 2015) and a common approach is to use gradient-based minimization algorithms, such as Gradient Descent (GD). This algorithm refines iteratively an initial solution $y_0 \in \mathcal{Y}$, by looking at the gradient of the function to be minimized, the main

	Positive definite matrix manifold (P^m_{++})	Sphere ($\mathcal{S}_{d-1}$)
$F(y)$	$\sum_{i=1}^{n} \alpha_i \| \log(Y^{-\frac{1}{2}} Z_i Y^{-\frac{1}{2}}) \|_F^2$	$\sum_{i=1}^{n} \alpha_i \arccos\left(\langle z_i, y \rangle\right)^2$
$\nabla_{\mathcal{M}} F(y)$	$2 \sum_{i=1}^{n} \alpha_i Y^{\frac{1}{2}} \log(Y^{\frac{1}{2}} Z_i^{-1} Y^{\frac{1}{2}}) Y^{\frac{1}{2}}$	$4 \sum_{i=1}^{n} \alpha_i (yy^T - I) \frac{\arccos(\langle z_i, y \rangle)}{\sqrt{1-\langle z_i, y \rangle}} z_i$
$R_y(v)$	$Y^{\frac{1}{2}} \exp(Y^{-\frac{1}{2}} v Y^{-\frac{1}{2}}) Y^{\frac{1}{2}}$	$\frac{v}{\|v\|}$

Table 3.1 Structured loss, gradient of the structured loss and retraction for P^m_{++} and S_{d-1}. $Z_i \in P^m_{++}$ and $z_i \in \mathcal{S}_{d-1}$ are the training set points. $I \in \mathbb{R}^{d \times d}$ is the identity matrix.

iteration can be written as

$$y_{t+1} = y_t - \eta_t \nabla F(y_t) \tag{3.12}$$

for a step size $\eta_t \in \mathbb{R}$. This algorithm can be extended to *Riemannian gradient descent* (RGD) (Zhang and Sra, 2016) on manifolds, as

$$y_{t+1} = exp_{y_t}(\eta_t \nabla_{\mathcal{M}} F(y_t)) \tag{3.13}$$

Where $\nabla_{\mathcal{M}} F$ is the gradient defined with respect to the Riemannian metric (Absil et al., 2009) and $exp_y : T_y\mathcal{M} \to \mathcal{M}$ denotes the exponential map on $y \in \mathcal{Y}$, mapping a vector from the tangent space $T_y\mathcal{M}$ to the associated point on the manifold according to the Riemannian metric of the manifold. For this family of gradient-based algorithms it is possible to substitute the exponential map with a retraction $R_y \colon T_y\mathcal{M} \to \mathcal{M}$, which is a first order approximation to the exponential map. Retractions are often faster to compute and still offer convergence guarantees (Absil et al., 2009). In the following experiments we will use both retractions and exponential maps. We mention that the step size η_t can be found with a line search over the validation set.

Table 3.1 reports gradients and retraction maps for the geodesic distance of two problems of interest considered in this work: positive definite manifold and the sphere. See Sections 3.5.2 and 3.5.3 for more details on the related manifolds.

We point out that the advantage of using optimization algorithms that comply with the geometry of the manifold, such as RGD, guarantees that the computed value is an element of the manifold. This is in contrast with algorithms that compute a solution in a linear space that contains $\mathcal{M}$ and then need to project the computed solution onto $\mathcal{M}$. Indeed

we will highlight this difference in the following experiments. We next discuss empirical evaluations of the proposed manifold structured estimator on both synthetic and real datasets.

3.5.2 Synthetic Experiments: Learning Positive Definite Matrices

We consider the problem of learning a function $f : \mathbb{R}^d \to \mathcal{Y} = P^m_{++}$, where P^m_{++} denotes the *cone of positive definite (PD)* $m \times m$ matrices, *i.e.* $P^m_{++} = \{M \in \mathbb{R}^{m\times m} : x^\top Mx > 0 \; \forall x \in \mathbb{R}^n\}$. Note that P^m_{++} is a Riemannian manifold (Bhatia, 2009; Moakher and Batchelor, 2006) with squared geodesic distance Δ_{PD} between any two PD matrices $Z, Y \in P^m_{++}$ defined as

$$\Delta_{\mathrm{PD}}(Z, Y) = \| \log(Y^{-\frac{1}{2}} Z \, Y^{-\frac{1}{2}}) \|^2_F$$

where, for any $M \in P^m_{++}$, we have that $M^{\frac{1}{2}}$ and $\log(M)$ correspond to the matrices with same eigenvectors of M but with, respectively, the square root and logarithm of the eigenvalues of M. In Table 3.1 we show the computation of the structured loss, the gradient of the structured loss and the exponential map of the PD matrix manifold.
For the experiments reported in the following, we compare the performance of the manifold structured estimator minimizing the loss Δ_{PD} and a Kernel Regularized Least Squares classifier (KRLS) baseline (detailed in Section 3.5.2), both trained using the Gaussian kernel $k(x, x') = \exp(-\|x - x'\|^2/2\sigma^2)$. The matrices predicted by the KRLS estimator are projected on the PD manifold by setting to a small positive constant $(1e-12)$ the negative eigenvalues. For the manifold structured estimator, the optimization problem at (3.8) was performed with the Riemannian Gradient Descent (RGD) algorithm (Absil et al., 2009).

KRLSs estimator for positive definite matrices

We consider the case where we want to use KRLS estimators to predict a positive definite matrix given a data set $\{x_i, y_i\}^n_{i=1}$ where $x_i, y_i \in P^d_{++} \; \forall i_1, \ldots, n$. The KRLS estimator $f : \mathbb{R}^d \to \mathbb{R}$ is a function defined as $f(x) = \sum^n_{i=1} k(x, x_i) w_i$, introduced in Chapter 1. To predict a positive definite matrix, we train a KRLS estimator for every element of the flattened matrix $vec(y_i) \in \mathbb{R}^{d^2}$. Suppose $j \in \{1, \ldots, d^2\}$ is the index of the j-th component of $vec(y)$ that we want to predict, then we learn the estimator $f^{(j)}(x) = \sum^n_{i=1} k(x, x_i) w^{(j)}_i$. The corresponding problem has labels $\hat{y}^{(j)} = [vec(y_1)_j, \ldots, vec(y_{d^2})_j]$ and we solve for $w^{(j)} = [w^{(j)}_1, \ldots, w^{(j)}_n]$. Indeed we learn a number of estimators equal to d^2, to predict $vec(f) = [f^{(1)}(x), \ldots, f^{(d^2)}]$ and then recover y from its vectorized form. Once the matrix is predicted we enforce it

	Squared loss		Δ_{PD} **loss**	
Dim	**KRLS**	**MSP**	**KRLS**	**MSP**
5	$0.72_{\pm 0.08}$	$0.89_{\pm 0.08}$	$111_{\pm 64}$	$0.94_{\pm 0.06}$
10	$0.81_{\pm 0.03}$	$0.92_{\pm 0.05}$	$44_{\pm 8.3}$	$1.24_{\pm 0.06}$
15	$0.83_{\pm 0.03}$	$0.91_{\pm 0.06}$	$56_{\pm 10}$	$1.25_{\pm 0.05}$
20	$0.85_{\pm 0.02}$	$0.91_{\pm 0.03}$	$59_{\pm 12}$	$1.33_{\pm 0.03}$
25	$0.87_{\pm 0.01}$	$0.91_{\pm 0.02}$	$72_{\pm 9}$	$1.44_{\pm 0.03}$
30	$0.88_{\pm 0.01}$	$0.91_{\pm 0.02}$	$67_{\pm 7.2}$	$1.55_{\pm 0.03}$

Table 3.2 Simulation experiment: average squared loss (First two columns) and Δ_{PD} (Last two columns) error of the proposed structured prediction (SP) approach and the KRLS baseline on learning the inverse of a PD matrix for increasing matrix dimension.

to be positive definite by performing a spectral decomposition and setting the negative eigenvalues to a small positive constant.

Learning the Inverse of a positive definite matrix.

We consider the problem of learning the function $f : P^m_{++} \rightarrow P^m_{++}$ such that $f(X) = X^{-1}$ for any $X \in P^m_{++}$. Input matrices are generated as $X_i = U\Sigma U^\top \in P^m_{++}$ with U a random orthonormal matrix sampled from the Haar distribution (Diestel and Spalsbury, 2014) and $S \in P^m_{++}$ a diagonal matrix with entries randomly sampled from the uniform distribution on $[0, 10]$. We generated datasets of increasing dimension m from 5 to 50, each with 1000 points for training, 100 for validation and 100 for testing. The kernel bandwidth σ was chosen and the regularization parameter λ were selected by cross-validation respectively in the ranges 0.1 to 1000 and 10^{-6} to 1 (logarithmically spaced).
Table 3.2 reports the performance of the manifold structured estimator (SP) and the KRLS baseline with respect to both the Δ_{PD} loss and the least squares loss (normalized with respect to the number of dimensions). Note that the KRLS estimator target is to minimize the least squares (Frobenius) loss and is not designed to capture the geometry of the PD cone. We notice that the proposed approach significantly outperforms the KRLS baseline with respect to the Δ_{PD} loss. This is expected: Δ_{PD} penalizes especially matrices with very different eigenvalues and our method cannot predict matrices with non-positive eigenvalues, as opposed to KRLS which computes a linear solution in $\mathbb{R}^{d^2}$ and then projects it onto the manifold. However the two methods perform comparably with respect to the squared loss. This is consistent with the fact that our estimator is aware of the natural structure of the output space and uses it profitably during learning.

	Δ Deg.
KRLS	26.9 ± 5.4
MR(Steinke et al., 2010)	22 ± 6
MSP (ours)	$\mathbf{18.8 \pm 3.9}$

Table 3.3 Fingerprints reconstruction: Average absolute error (in degrees) for the manifold structured estimator (MSP), the manifold regression (MR) approach in (Steinke et al., 2010) and the KRLS baseline.

3.5.3 Fingerprint Reconstruction

We consider the fingerprint reconstruction application introduced by Steinke et al. (2010) in the context of manifold regression. Given a partial image of a fingerprint, the goal is to reconstruct the contour lines in output. Each fingerprint image is interpreted as a separate structured prediction problem where training input points correspond to the 2D position $x \in \mathbb{R}^2$ of valid contour lines and the output is the local orientation of the contour line, interpreted as a point on the circumference $\mathcal{S}_1$. The space $\mathcal{S}_1$ is a manifold with squared geodesic distance $\Delta_{\mathcal{S}_1}$ between two points $z, y \in \mathcal{S}_1$ corresponding to

$$\Delta_{\mathcal{S}_1}(z, y) = \arccos\left(\langle z, y \rangle\right)^2$$

where arccos is the inverse cosine function. In Table 3.1 we show the computation of the structured loss, the gradient of the structured loss and the chosen retraction for the sphere manifold. We compared the performance of the manifold structured estimator proposed in this chapter with the manifold regression approach introduced by Steinke et al. (2010) on the FVC fingerprint verification challenge dataset[2]. The dataset consists of 48 fingerprint pictures, each with ~ 1400 points for training, ~ 1000 points for validation and the rest (~ 25000) for test.

Figure 3.2 reports the average absolute error (in degrees) between the true contour orientation and the one estimated by our structured prediction approach (SP), the manifold regression (MR) proposed by Steinke et al. (2010) and the KRLS baseline. Our method outperforms the MR competitor by a significant margin. As expected, the KRLS baseline is not able to capture the geometry of the output space and has a significantly larger error of the two other approaches. This is also observed on the qualitative plot in Figure 3.2 (Left) where the predictions of our SP approach and the KRLS baseline are compared with the ground truth on a single fingerprint. Output orientations are reported for each pixel with

[2] http://bias.csr.unibo.it/fvc2004

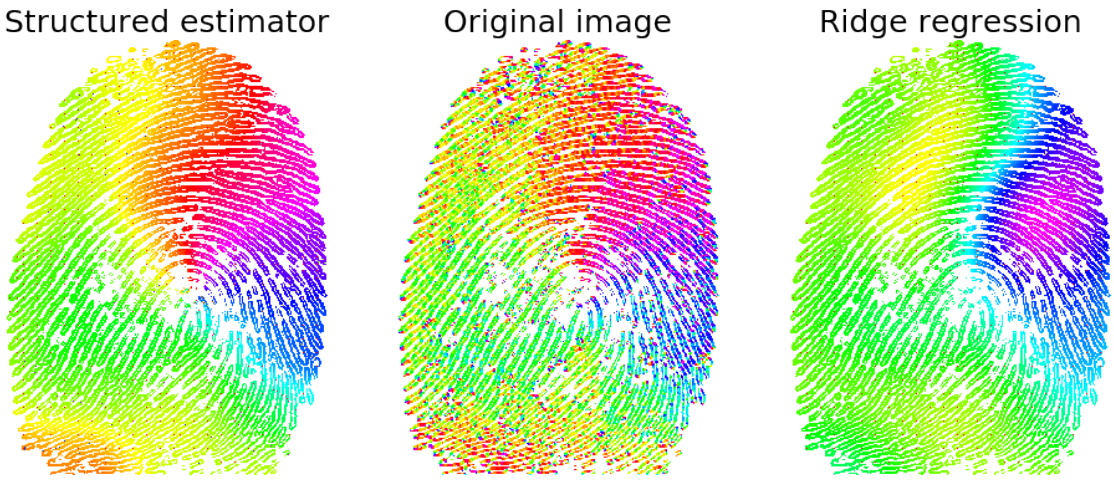


Figure 3.2 Fingerprint reconstruction of a single image where the structured predictor achieves 15.7 of average error while KRLS 25.3.

a color depending on their orientation (from 0 to π). While the KRLS predictions are quite inconsistent, it can be noticed that our estimator is very accurate and even "smoother" than the ground truth.

3.5.4 Multilabel Classification on the Statistical Manifold

We evaluated our algorithm on multilabel prediction problems. In this context the output is an m-dimensional histogram, i.e. a discrete probability distribution over m points. We consider as manifold the space of probability distributions over m points, that is the m-dimensional simplex Δ^m endowed with the *Fisher information metric* (Amari and Nagaoka, 2007). We will consider $\mathcal{Y} = \Delta_\epsilon^m$ where we require $y_1, \ldots, y_m \geq \epsilon$, for $\epsilon > 0$. In the experiment we considered $\epsilon = 1e-5$. The geodesic induced by the Fisher metric is, $d(y, y') = \arccos\left(\sum_{i=1}^m \sqrt{y_i y'_i}\right)$ (Nielsen and Sun, 2017). This geodesic comes from applying the map $\pi\colon \Delta^m \to \mathcal{S}_{m-1}$, $\pi(y) = (\sqrt{y_1}, \ldots, \sqrt{y_{m+1}})$ to the points $\{y_i\}_{i=1}^n \in \Delta^m$. This results in points that belong to the intersection of the positive quadrant $\mathbb{R}_{++}^m$ and the sphere $\mathcal{S}_{m-1}$. We can therefore use the geodesic distance on the Sphere and gradient and retraction map described in Table 3.1. We test our approach on some of the benchmark multilabel datasets described in (Tsoumakas et al., 2009) and we compare the results with the KRLS baseline. We cross-validate λ and σ taking values, respectively, from the intervals $[1e-6, 1e-1]$ and $[0.1, 10]$. We compute the area under curve (AUC) (Srinivasan, 1999) metric to evaluate the quality of the predictions, results are shown in Table 3.4. It can be observed that for the easier dataset of CAL500 there's no gain by taking into account

	AUC	
	vKRLS	MSP (Ours)
Emotions	0.63	**0.73**
CAL500	**0.92**	**0.92**
Scene	0.62	**0.73**

Table 3.4 Area under the curve (AUC) on multilabel benchmark datasets (Tsoumakas et al., 2009) for KRLS and SP.

the structure of the manifold. however a difference in performance is clear on the more challenging datasets, Emotions and Scene.

In this chapter we showed how it is possible to perform consistent regression on dense structured spaces such as differentiable manifold. And how, even when considering naïve approaches that operate in the ambient Euclidean spaces, taking into account the structure of the manifold is strongly beneficial to the final result of the prediction.

Chapter 4

Hyperbolic Data Representation

4.1 Introduction

Representation learning is a key paradigm in machine learning and artificial intelligence. It has enabled important breakthroughs in computer vision (He et al., 2016; Krizhevsky et al., 2012) natural language processing (Bojanowski et al., 2016; Joulin et al., 2016; Mikolov et al., 2013), relational learning (Nickel et al., 2011; Perozzi et al., 2014), generative modeling (Kingma and Welling, 2013; Radford et al., 2015), and many other areas (Bengio et al., 2013; LeCun et al., 2015). Its objective is typically to infer latent feature representations of objects (e.g., images, words, entities, concepts) such that their similarity or distance in the representation space captures their *semantic* similarity. For this purpose, the geometry of the representation space has recently received increased attention (Davidson et al., 2018; Falorsi et al., 2018; Wilson et al., 2014; Xu and Durrett, 2018). In this chapter we focus on Riemannian representation spaces and in particular on hyperbolic geometry. Nickel and Kiela (2017) introduced Poincaré embeddings to infer hierarchical representations of symbolic data and showed that they lead to substantial gains in representational efficiency and generalization performance. Hyperbolic representations have since been extended to further manifolds (De Sa et al., 2018; Nickel and Kiela, 2018), word embeddings (Le et al., 2019; Tifrea et al., 2018), recommender systems (Chamberlain et al., 2019), and image embeddings (Khrulkov et al., 2019).

However, it is yet an open problem how to efficiently integrate hyperbolic representations with standard machine learning methods which often make a Euclidean or vector space assumption. In the work by Ganea et al. (2018) some fundamental steps have been made in this direction by proposing a generalization of fully connected neural network layers from Euclidean space to hyperbolic space. However most of the experiments shown were

from hyperbolic to Euclidean space using recurrent models. Therefore in this chapter we focus on the task of learning manifold-valued functions from Euclidean on to hyperbolic space that allows us to leverage its hierarchical structure for supervised learning. For this purpose, we propose two novel approaches: a non-parametric kernel-based model – for which we also prove generalization bounds – and a parametric deep learning model which is informed by the geodesics of the output space to learn hyperbolic-valued functions.
We illustrate the efficacy of our approach on two challenging tasks, i.e., hierarchical classification via label embeddings and taxonomy expansion by predicting concept embeddings from text. For standard classification tasks, label embeddings have shown great promise as they allow to scale supervised learning methods to datasets with massive label spaces (Chollet, 2016; Veit et al., 2018). By embedding labels in hyperbolic space according to their natural hierarchical structure (e.g, the underlying WordNet taxonomy of ImageNet labels) we are then able to combine the benefits of hierarchical classification with the scalability of label embeddings. Moreover, the continuous nature of hyperbolic space allows us to *invent* new concepts by predicting their placement in a pre-embedded base taxonomy. We exploit this property for a novel task which we refer to as *taxonomy expansion*: Given an embedded taxonomy $\mathcal{T}$, we show that we can infer the correct placement of unknown concepts by learning to regress their Euclidean features onto the embedding. In contrast to hierarchical classification, the predicted embeddings are here full members of the taxonomy, i.e., they can themselves act as parents of other points. For both tasks we show empirically that our proposed methods achieve strong generalization performance and outperform their Euclidean regression counterparts. Moreover we also investigate empirical results of hyperbolic neural networks (Ganea et al., 2018) compared to our methods in the task of taxonomy expansion, showing that it is not necessary to work with hyperbolic layers as long as the training procedure exploits the geodesic as an error measure.
The remainder of this chapter is organized as follows: in Section 4.2 we introduce hyperbolic embeddings and related concepts such as Riemannian optimization. In Section 4.3, we introduce our proposed methods and prove excess risk bounds for the kernel-based method. In Section 4.5 we evaluate our methods on the tasks of hierarchical classification and taxonomy expansion.

4.2 Hyperbolic Manifold

In Chapter 3 we already introduced basic concepts of differentiabl geometry and some notable manifolds. Because this chapter is solely concerned with the hyperbolic manifold, we devote the following section to introducing this space and its characteristics. Hyperbolic space is the unique, complete, simply connected Riemannian manifold with constant negative sectional curvature. There exist multiple equivalent models for hyperbolic space that offer different geometric interpretation of such space. These models have seen various uses in fields such as physics where they are used to formalize some models of special relativity (Ungar, 2008), and more recently in network science, where they used to model complex network topologies. Hyperbolic space is well suited for representing hierarchical data due to its geometry. By placing the root node at the origin of hyperbolic space, the manifold geometry replicates a tree-like structure. We will now give two examples substantiating this intuition, for a more in depth discusssion we refer the reader to the work of Hamann (2018).

Consider the task of representing a tree into a metric space, such that its structure is reflected in the embedding. A regular tree with branching factor b has $(b+1)b^{l-1}$ nodes at level l. The number of nodes grows exponentially with respect to their distance from the root. On the other hand, if we consider a circle with radius r in hyperbolic space, its length grows *exponentially* with respect to r. This is false in the case of Euclidean space where the length of any circumference grows only quadratically with respect to its radius. Another example: in a graph the shortest path between two nodes x, y that have a common node O, passes through that common node, *i.e.* $d(x,y) = d(x,O) + d(y,O)$ so the ratio between their distance and the sum of their distances from O is $\frac{d(x,y)}{d(x,O)+d(O,y)} = 1$. In hyperbolic space, as two points x, y get farther from the origin, the ratio between their distance and the sum of their distances from the origin asymptotically approaches 1, *i.e.* $\lim_{\|y\|,\|x\|\to\infty} \frac{d(x,y)}{d(x,O)+d(O,y)} = 1$, this means that it is possible to arbitrarily approximate this distance. This is not true for euclidean space where the ratio is constant. A representation of this phenomenon is depicted in Figure 4.1.

Two common models for hyperbolic manifold are the Lorentz model and the Poincaré model. We will use the former to estimate embeddings using stochastic optimization due to its numerical advantages. For analysis, we will map embeddings into the Poincaré disk which provides an intuitive visualization of hyperbolic embeddings. Figure 4.2 shows an example of tree embedding in the Poincaré model.

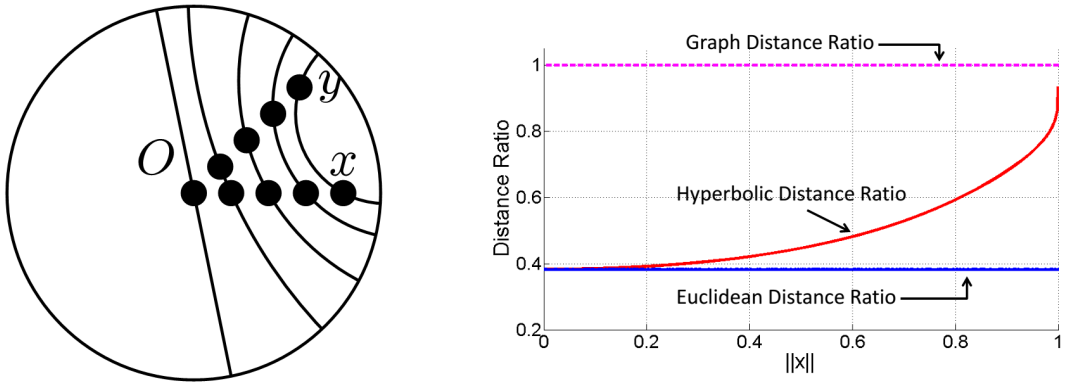


Figure 4.1 Shortest path between two points in hyperbolic space and the ration between their distance and the distance from the origin.

4.2.1 Lorentz Model

Let $y, z \in \mathbb{R}^{d+1}$ and let $\langle y, z \rangle_{\mathcal{L}} = -y^{(0)}z^{(0)} + \sum_{i=1}^{d} y^{(i)}z^{(i)}$ denote the *Lorentzian scalar product*. The Lorentz model of d-dimensional hyperbolic space is then defined as the Riemannian manifold $\mathcal{L}^d = (\mathcal{M}^d, g_{\mathcal{L}})$, where

$$\mathcal{M}^d = \{y \in \mathbb{R}^{d+1} : \langle y, y \rangle_{\mathcal{L}} = -1, y^{(0)} > 0\}, \tag{4.1}$$

denotes the upper sheet of a two-sheeted n-dimensional hyperboloid and where $g_{\mathcal{L}}(y) = \text{diag}([-1, 1, \ldots, 1])$ is the associated metric tensor.
Furthermore, the distance on $\mathcal{L}$ is defined as

$$d_{\mathcal{L}}(y, z) = \text{acosh}(-\langle y, z \rangle_{\mathcal{L}}). \tag{4.2}$$

An advantage of the Lorentz model is that its exponential map has as simple, closed-form expression. As Nickel and Kiela (2018) showed, this allows us to perform Riemannian optimization efficiently and with increased numerical stability. In particular, let $y \in \mathcal{L}^d$ and let $t \in \mathcal{T}_y\mathcal{L}^d$ denote a point in the associated tangent space. The exponential map $\exp_y : \mathcal{T}_y\mathcal{L}^d \to \mathcal{L}^d$ is then defined as

$$\exp_y(t) = \cosh(\|t\|_{\mathcal{L}})y + \sinh(\|t\|_{\mathcal{L}})\frac{t}{\|t\|_{\mathcal{L}}}. \tag{4.3}$$

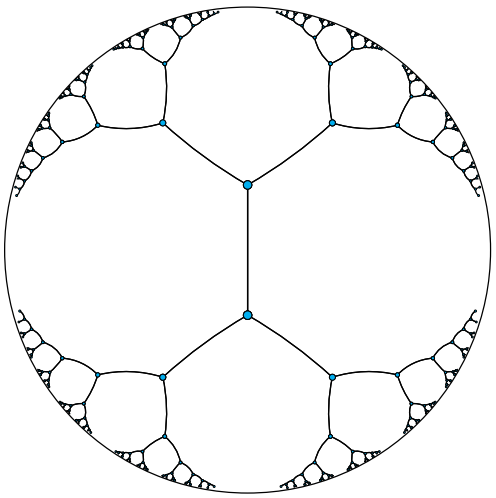


Figure 4.2 Embedding of a tree in the Poincare disk.

Because the Lorentz model of hyperbolic space is a smooth sub-manifold of Euclidean space (Robbin and Salamon, 2011), the gradient of any smooth function $\mathcal{F}\colon \mathcal{M}^d \to \mathbb{R}$ at a point $y \in \mathcal{M}^d$ can be computed by taking the direction of steepest ascent h and projecting it onto the tangent space of y Absil et al. (2009). The direction of steepest ascent can be computed from the Euclidean gradient

$$h = g_{\mathcal{L}}^{-1} \nabla f(y) \tag{4.4}$$

while the projection $\text{proj}_y\colon \mathbb{R}^{d+1} \to \mathcal{T}_y\mathcal{M}^d$, for the Lorentz model has form

$$\text{proj}_y(h) = h - \frac{\langle y, h\rangle_{\mathcal{L}}}{\langle y, y\rangle_{\mathcal{L}}} y = h + \langle y, z\rangle_{\mathcal{L}} h \tag{4.5}$$

since $\forall\, y \in \mathcal{L}^d$ it is that $\langle y, y\rangle_{\mathcal{L}} = -1$.

4.2.2 Poincaré ball

The Poincaré ball model is the Riemannian manifold $\mathcal{P}^d = (\mathcal{B}^d, g_p)$, where

$$\mathcal{B}^d = \{y \in \mathbb{R}^d : \|y\| < 1\} \tag{4.6}$$

is the *open* n-dimensional unit ball and where $g_p(y) = 4/(1-\|y\|^2)^2$ is the associated metric tensor. The distance function on $\mathcal{P}$ is defined as

$$d_p(y,z) = \operatorname{acosh}\left(1+2\frac{\|y-z\|^2}{(1-\|y\|^2)(1-\|z\|^2)}\right). \tag{4.7}$$

An example of geodesics in the Poincaré model is depicted in Figure 4.3a
An advantage of the Poincaré ball is that it provides an intuitive model of hyperbolic space which is well suited for analysis and visualization of the embeddings. It can be seen from Equation (4.7), that the distance within the Poincaré ball changes smoothly with respect to the norm of y and z. This locality property of the distance is key for representing hierarchies efficiently. For instance, by placing the root node of a tree at the origin of $\mathcal{B}^n$, it would have relatively small distance to all other nodes, as its norm is zero. On the other hand, leaf nodes can be placed close to the boundary of the ball, as the distance between points grows quickly with a norm close to one.

4.2.3 Isometries

The Lorentz and Poincaré ball are equivalent in that there exist isometric mappings $p : \mathcal{L}^d \to \mathcal{P}^d$ and $p^{-1} : \mathcal{P}^d \to \mathcal{L}^d$ between both manifolds, which are given as

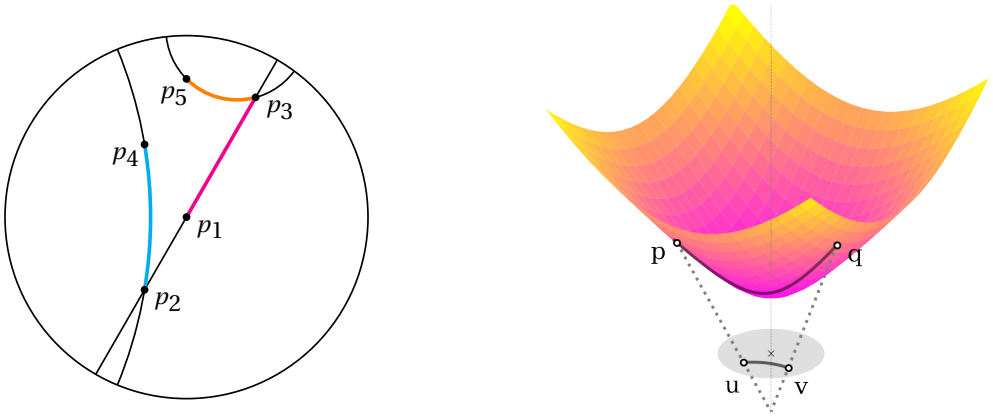
$$p(y^{(0)}, y^{(1)}, \ldots, y^{(d)}) = \frac{(y^{(1)}, \ldots, y^{(d)})}{y^{(0)}+1}; \tag{4.8}$$

$$p^{-1}(y^{(1)}, \ldots, y^{(d)}) = \frac{(1+\|y\|^2, 2y^{(1)}, \ldots, 2y^{(d)})}{1-\|y\|^2}. \tag{4.9}$$

This allows us to freely map between the two hyperbolic models, e.g., to perform optimization in $\mathcal{L}$ and analyze the learned embeddings in $\mathcal{P}$.

4.3 Hyperbolic representation

In machine learning, data representation aims at finding suitable transformations for the input data, such that the algorithms at hand can operate on the transformed data with increased performance, be it efficiency or accuracy. A typical example of this are feature maps, that map input data into higher dimensional linear spaces where it is easier to find separate data with linear functions.



(a) (b)

Figure 4.3 a) Geodesics in the Poincaré disk model of hyperbolic space. b) Lorentz model of hyperbolic space and its mapping onto the Poincaré disk.

Embeddings are representations used for structured data like text or graphs, these transformations map atomic elements from the starting dataset, such as pair of words or graph nodes, into points of a linear space thus allowing for classical machine learning algorithms, such as support vector machines and deep models, to perform regression or classification. However, for many problems, the label space is naturally endowed with a hierarchical structure – the underlying WordNet taxonomy of ImageNet labels being a prime example (Deng et al., 2009; Strapparava et al., 2004). We leverage such structure by embedding labels with hierarchical structure in hyperbolic space and transforming a classification problem into a manifold regression problem, albeit in a non-linear space. By learning to predict the correct embedding from Euclidean features we are then able to combine the benefits of hierarchical classification with the scalability of label embeddings.

4.3.1 Hyperbolic representation

We consider supervised datasets $\{x_i, c_i\}_{i=1}^m \in \mathcal{X} \times \mathcal{C}$ where class labels c_i can be organized according to a taxonomy or class hierarchy $\mathcal{T} = (\mathcal{C}, \mathcal{E})$. Edges $(i, j) \in \mathcal{E}$ indicate the ontological relation c_i *is-a* c_j. The goal is to find the set $\Theta = \{y_i\}_{i=1}^{|\mathcal{C}|}$ which is the set of points $y_i \in \mathcal{M}^d \; \forall i = 1, \ldots, |\mathcal{C}|$ that correspond to the nodes of $\mathcal{T}$ in hyperbolic space, mantaining the structure structure of the original tree. To compute hyperbolic embeddings y_i of all c_i that capture these hierarchical relationships of $\mathcal{T}$, we follow Nickel and Kiela (2017, 2018) and infer the embedding from pairwise similarities. In particular, let $\gamma : \mathcal{C} \times \mathcal{C} \to \mathbb{R}_+$ be the

similarity function such that

$$\gamma(c_i, c_j) = \begin{cases} 1, & \text{if } c_i, c_j \text{ are adjacent in } clos(\mathcal{T}) \\ 0, & \text{otherwise} \end{cases} \tag{4.10}$$

where $clos(\mathcal{T})$ is the transitive closure of $\mathcal{T}$. Furthermore, let $\mathcal{N}(i,j) = \{\ell : \gamma(i,\ell) < \gamma(i,j)\} \cup \{j\}$ denote the set of concepts that are *less* similar to c_i then c_j (including c_j) and let $\phi(i,j) = \arg\min_{k \in \mathcal{N}(i,j)} d(y_i, y_k)$ denote the nearest neighbor of c_i in the set $\mathcal{N}(i,j)$. We then learn embeddings $\Theta = \{y_i\}_{i=1}^m$ by optimizing

$$\min_{\Theta} -\sum_{i,j} \log \Pr(\phi(i,j) = j \mid \Theta) \tag{4.11}$$

with

$$\Pr(\phi(i,j) = j \mid \Theta) = \frac{e^{d(y_i, y_j)}}{\sum_{k \in \mathcal{N}(i,j)} e^{d(y_i, y_k)}}. \tag{4.12}$$

Equation (4.12) can be interpreted as a ranking loss that aims to extract latent hierarchical structures from $\mathcal{C}$ (Nickel and Kiela, 2018). For computational efficiency, we follow Jean et al. (2014) and randomly subsample $\mathcal{N}(i,j)$ on large datasets. To infer the embeddings θ we then minimize Equation (4.12) using Riemannian SGD as introduced in Section 3.5.1 We apply the transitive closure in Equation (4.10) following the intuition in *Poincaré Embeddings for Learning Hierarchical Representations* (Nickel and Kiela, 2017). We mentioned in Section 4.2 that the ratio between the distances of two points their distance and the sum of theirs distances from the origin approaches 1. This effectively implies that the origin of Hyperbolic space is closer to those points than they are one from each other. Since we compute the embeddings using Equation (4.12) without looking at the actual node distances in the tree but only checking if two nodes are connected, the transitive closure avoids the embedding of a node to be far from its descendants.

By computing hyperbolic embeddings of $\mathcal{T}$, we have then recast the learning problem from a discrete tree $\mathcal{D} = \{x_i, c_i\}_{i=1}^m, c_i \in \mathcal{C}$ to its embedding in a continuous manifold $\mathcal{D}^e = \{x_i, y_i\}_{i=1}^m$ with $y_i \in \mathcal{L}$. This allows us to apply manifold regression techniques to perform classification and even learn new concepts by learning their corresponding embedding in the manifold. To incorporate hierarchy information in the label embedding, we also study regressing onto an *augmented hierarchy* where we add each example x_i as a child to its associated class c_i in $\mathcal{T}$. We then embed this augmented hierarchy $\mathcal{T}_x$ using hyperbolic embeddings as described in Section 4.2. However, we compute similarity

scores $\gamma(\cdot,\cdot)$ in the transitive closure of $\mathcal{T}_x$ using a Gaussian kernel when features for both nodes are available or the original γ otherwise.
We use the *augmented hierarchy* to overcome the over-simplified original class hierarchy. The embedding learning process leverages also similarity among the features samples, leading to a richer embedding. In our experiments we verified empirically that following this strategy, similar samples from the same class tend to be closer in hyperbolic space. Hence, different classes tend to form separate clusters of neighboring points in hyperbolic space. In particular, the clusters associated to classes with similar semantic content tend to be closer in hyperbolic space. Using the augmented hierarchy, the computed embedding achieved higher fidelity with respect to the original structure.

4.4 Hyperbolic regression

Having introduced how to compute hyperbolic representations the next natural question is how to learn We consider the problem of manifold regression detailed in Chapter 3. We assume for simplicity that $\mathcal{X} \subset \mathbb{R}^d$ and $\mathcal{Y} \subset \mathcal{L}^n$ are compact subsets. Here we consider $\mathcal{L}^n$ as target space and $\Delta(\cdot,\cdot) = d_{\mathcal{L}}(\cdot,\cdot)^2$ as loss function, but all results extend to $\mathcal{P}$. We tackle this problem proposing two approaches: one leveraging the results of Chapter 3 and one using geodesic neural networks.

4.4.1 Hyperbolic Structured Prediction

In Chapter 3 we proposed a new approach to address manifold regression problems. We adopted a perspective based on structured prediction and interpreted the target manifold $\mathcal{Y}$ as a "structured" output. We formulate the corresponding Hyperbolic Structured Prediction (HSP) estimator when applying this strategy to our problem (namely $\mathcal{Y} \subset \mathcal{L}^n$). In particular, we have $f_{hsp} : \mathcal{X} \to \mathcal{L}^n$ the function such that for any test point $x \in \mathcal{X}$

$$f_{hsp}(x) = \operatorname*{arg\,min}_{y \in \mathcal{L}^n} \sum_{i=1}^{m} \alpha_i(x)\, d_{\mathcal{L}}(y, y_i)^2, \tag{4.13}$$

where the weights $\alpha(x) = (\alpha_1(x), \ldots, \alpha_n(x))^\top \in \mathbb{R}^m$ are learned according to Equation (3.9). We studied the generalization properties of the estimator in Chapter 3. We proved under suitable assumptions on the regularity of the output manifold, that it was possible to give bounds on the excess risk in terms of the number of training examples available. The following result specializes Theorem 3.4.3 to the case of f_{hsp}. A key role will be played by

the $(s,2)$-Sobolev space $W^{s,2}$ of functions from $\mathcal{L}^n$ to $\mathbb{R}$, which generalizes the standard notion on Euclidean domains (see Hebey (2000) for an introduction on the topic).

Theorem 4.4.1. *Let $(x_i, y_i)_{i=1}^m$ be sampled independently according to ρ on $\mathcal{X} \times \mathcal{Y}$ with $\mathcal{Y} \subset \mathcal{L}^m$ a compact subset. Let f_{hsp} defined as in Equation (4.13) with weights from Equation (3.9) learned with reproducing kernel $k : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$ with reproducing kernel Hilbert space (RKHS) $\mathcal{F}$. If the map $x \mapsto \int d_{\mathcal{L}}(\cdot, y)^2 \, d\rho(y|x)$ belongs to $W^{s,2}(\mathcal{Y}) \otimes \mathcal{F}$ with $s > n/2$, then for any $\tau \in (0,1]$*

$$\mathcal{E}(f_{hsp}) - \inf_f \mathcal{E}(f) \;\leq\; \left\| d_{\mathcal{L}}^2 \right\|_{s,2} \mathsf{q} \tau^2 \, \frac{1}{n^{1/4}}, \tag{4.14}$$

holds with probability at least $1 - 8e^{-\tau}$, where q is a constant not depending on n, τ or $\left\| d_{\mathcal{L}} \right\|_{s,2}$.

The proof of the result above hinges on Theorem 3.4.3 and is detailed in the next subsection. The result guarantees a learning rate of order $O(n^{-1/4})$. We comment on the assumptions and constants appearing in Theorem 4.4.1. First, we point out that, albeit the requirement $\int d_{\mathcal{L}}(\cdot, y)^2 \, d\rho(y|x) \in \mathcal{F} \otimes W^{s,2}(\mathcal{Y})$ can seem overly abstract, it reduces to a standard assumption in statistical learning theory. Informally, it corresponds to a regularity assumption on the conditional mean embedding of the distribution $\rho(\cdot|x)$ (see e.g. (Song et al., 2013) for more details), and can be interpreted as requiring the solution of Equation (3.7) to belong to the hypotheses space associated to the kernel k. Second, we comment on the constant in Equation (4.14) that depending on the geodesic distance. In particular, we note that by Theorem 3.4.2 the squared geodesic on any compact subset of $\mathcal{L}^n$ belongs to $W^{s,2}(\mathcal{Y})$ for any $s \geq 0$. Hence $\left\| d_{\mathcal{L}}^2 \right\|_{s,2} < +\infty$ also for any $s > n/2$, as required by Theorem 4.4.1.

Bound proof

The proof of Theorem 4.4.1 is a specialization of Theorems 3.4.2 and 3.4.3. We recall a key assumption that is required to apply such results.

Assumption 4.4.1. *$\mathcal{M}$ is a complete n-dimensional smooth connected Riemannian manifold, without boundary, with Ricci curvature bounded below and positive injectivity radius.*

The assumption above imposes basic regularity conditions on the output manifold. A first implication is the following.

Proposition 4.4.1 (Theorem 3.4.2)**.** *Let $\mathcal{M}$ satisfy Assumption 4.4.1 and let $\mathcal{Y} \subset \mathcal{M}$ is a compact geodesically convex subset of $\mathcal{M}$. Then, the squared geodesic distance $d : \mathcal{M} \times$*

$\mathcal{M} \to \mathbb{R}$ is smooth on $\mathcal{Y}$. Moreover, by the proof of Theorem 3.4.1 in Section A.1, we have $d^2 \in W^{s,2}(\mathcal{Y})$ for any $s > n/2$.

Leveraging standard results from Riemannian geometry, we can guarantee that the manifolds considered in this chapter satisfy the above requirements. For simplicity, we restrict on $\mathcal{M}$ corresponding to an open bounded ball in either $\mathcal{P}^n$ or $\mathcal{L}^n$. In particular,

- $\mathcal{M}$ has sectional curvature constantly equal to -1. Hence the Ricci curvature is bounded from below since we are in a bounded ball in either $\mathcal{P}^n$ or $\mathcal{L}^n$.
- The injectivity radius is positive (actually lower bounded by $1/(2 \cdot 9^{2+[n/2]})$ with $[n/2]$ the integer parts of $n/2$), see Main Theorem in Martin (1989).

We see that we are in the hypotheses of Proposition 4.4.1, from which we conclude the following.

Corollary. *For any $s \geq 0$, the geodesic distance $d_{\mathcal{L}}$ (respectively $d_{\mathcal{P}}$) belongs to $W^{s,2}(\mathcal{Y})$ for any compact subspace of $\mathcal{L}^n$ (respectively $\mathcal{P}^n$).*

This guarantees us that we are in the hypotheses of Theorem 3.4.3, from which Theorem 4.4.1 follows immediately. We note in particular that the corollary guarantees that $d^2_{\mathcal{L}}$ is SELF, as defined in Definition 2.2.1.

4.4.2 Homeomorphic Geodesic Neural Network

As an alternative to the non-parametric model f_{hsp}, we propose also a parametric method based on deep neural networks. An important challenge when dealing with manifold regression is how to design a suitable model for the estimator. While neural networks of the form $g_w \colon \mathbb{R}^d \to \mathbb{R}^k$ (parametrized by some weights w) have proven to be powerful models for regression and feature representation (Bengio et al., 2013; LeCun et al., 2015; Ngiam et al., 2011; Xiao et al., 2016), the study of these models for learning manifold-valued function is limited. Some recent works have investigated how to define layers that generalize classical neural network layers in hyperbolic space (Ganea et al., 2018; Tay et al., 2017), however the focus has been on applying these models to learn maps from hyperbolic manifolds to Euclidean space. On the other hand, in this chapter we consider the Poincaré model and develop a neural architecture mapping the Euclidean space into the open unit ball.

Consider the element-wise hyperbolic tangent be defined as

$$t\colon \mathbb{R}^k \to \{x \in \mathbb{R}^k : \|x\|_\infty < 1\} \tag{4.15}$$

$$(x^{(1)}, \ldots, x^{(k)}) \mapsto (\tanh x^{(1)}, \ldots, \tanh x^{(k)}), \tag{4.16}$$

which maps a linear space onto the open ℓ_∞ ball. Moreover, we define a "squashing" function

$$s\colon \{x \in \mathbb{R}^k : \|x\|_\infty < 1\} \to \{x \in \mathbb{R}^k : \|x\|_2 < 1\} \tag{4.17}$$

$$s(x) = \begin{cases} x \mapsto x \frac{\|x\|_\infty}{\|x\|_2}, & \text{if } x \neq \mathbf{0} \\ \mathbf{0}, & \text{if } x = \mathbf{0} \end{cases} \tag{4.18}$$

where $\mathbf{0}$ is the vector of all zeros. Because $\|x\|_\infty < \|x\|_2$, s is continuous and maps the open ℓ_∞ ball into the open ℓ_2 ball. And because both s and t are bijective continuous function with continuous inverse, the composition $s \circ t\colon \mathbb{R}^k \to \{x \in \mathbb{R}^k : \|x\|_2 < 1\}$ is a homeomorphism from $\mathbb{R}^k$ into the open ball ℓ_2 and therefore also on the Poincaré model manifold. By composing $s \circ t$ with the neural network feature extractor g_w we obtain a deep model that jointly learns features into a linear space and maps them to the hyperbolic manifold:

$$f_{nng} = s \circ t \circ g_w\colon \mathbb{R}^d \to \mathcal{P}^k. \tag{4.19}$$

Here g_w corresponds to the enconding function e_w and $s \circ t$ is the decoding function, where no explicit maximization over y is present. Since the only trainable parameters are the weights w in the feature extractor g_w, and the decoding $s \circ t$ is sub-differentiable, the training of this model is akin to training a classical deep learning architecture. An operation that can be easily implemented with common deep learning frameworks such as PyTorch (Paszke et al., 2017) or TensorFlow (Abadi et al., 2016). However we train this model, using the squared geodesic instead of the classical Euclidean loss.

4.5 Applications

We have introduced two models for predicting values in hyperbolic space. Both methods, HSP and NN-G, benefit from label embeddings that reflect the similarity of the associated examples and exploit the hyperbolic structure of the output space.
We evaluate our proposed methods for hyperbolic manifold regression on the following tasks:

Hierarchical Classification via label embeddings. For this task, the goal is to classify examples with a single label from a class hierarchy with tree structure. We leverage hyperbolic representations for this task and first compute label embeddings of the class hierarchy. We then learn to regress examples onto label embeddings and classify them using the nearest label in the target space, i.e., by denoting $y_c \in \mathcal{L}^n$ the embedding of class c and taking $f : \mathbb{R}^d \to \mathcal{L}^n$.

$$\widehat{c} = \arg\min_{c \in \mathcal{C}} d_{\mathcal{L}}(f(x), y_c) \tag{4.20}$$

Taxonomy expansion. The goal is to expand an existing taxonomy based on feature information about new concepts. As for hierarchical classification, we first embed the existing taxonomy in hyperbolic space and then learn to regress onto the label embeddings. However, a key difference is that a new example c can act as the parent of another class c'.

4.5.1 Models and training details

For hierarchical classification, we compare to standard baselines such as top-down classification with logistic regression (TD-LR) and hierarchical SVM (HSVM). Furthermore, since both tasks can be regarded as regression problems onto the Poincaré ball (which has a canonical embedding in $\mathbb{R}^k$) we also compare to kernel regularized least squares regression (KRLS) and a neural network with squared Euclidean loss (NN-E). In both cases, we constrain predictions to remain within the Poincaré ball via the projection

$$\text{proj}(y) = \begin{cases} y/\|y\| - \varepsilon & \text{if } \|y\| \geq 1 \\ y & \text{otherwise} \end{cases},$$

where ε is a small constant to ensure numerical stability, equal to $\varepsilon = 10^{-6}$. These regression baselines allows us to evaluate the advantages of training manifold valued models with squared geodesic loss compared to standard methods that are agnostic of the underlying geodesics.

For kernel-based methods, we employ a Gaussian kernel selecting the bandwidth $\sigma \in [10^{-1}, 10^{2}]$ and regularization parameter $\lambda \in [10^{-6}, 10^{-2}]$ via cross-validation. Both parameter ranges are logarithmically spaced. For HSP inference we use RSGD with batch size equal to 50 and a maximum of 40000 iterations we stop the minimization if the the gradient Euclidean norm is smaller than 10^{-5} (In most cases the inference stops before the 10000 iteration). The learning rate for RSGD is chosen via cross-validation on the interval $[10^{-5}, 10^{-1}]$. For the neural network models (NN-G, NN-E) we use the same architecture

for g_w. Each layer is a fully connected layer where $\sigma(x) = \max(0, x)$ is a ReLU non-linearity and $w = \{W \in \mathbb{R}^{u_l \times 2u_l}\}$, *i.e.* each layer halves the number of hidden units with respect to the previous layer (with the exception of the first and last layer which must fit input and output dimensions). We use a depth of 5 layers with intermediate dimensionalities $u_l \in (1024, 1024, 512, 256, 128)$ for taxonomy expansion and $u_l \in (2048, 2048, 1024, 512, 256)$ for hierarchical classification. We did not find significant improvements with deeper architectures in performance. We train the deep models using mini-batch stochastic gradient descent, with a scheduler until the model reaches convergence of the training loss. For the WordNet Mammals dataset we also compare our algorithms with a hyperbolic neural network (HNN) as introduced by Ganea et al. (2018), this architecture is trained with Riemannian Gradient Descent until convergence and has the same structure and the same number of parameters of NN-G and NN-E.

4.5.2 Hierarchical classification

For hierarchical classification, we are given a supervised training set $\mathcal{D} = \{x_i, c_i\}_{i=1}^m$ where the class labels c_i are organized in a tree $\mathcal{T}$. We first embed the *augmented hierarchy* $\mathcal{T}_x$ as discussed in Section 4.3.1 and learn a regression function $\hat{f}\colon \mathbb{R}^d \to \mathcal{L}^n$ using $\mathcal{D}^e = \{x_i, y_i\}_{i=1}^m$. For a test point $x' \in \mathbb{R}^d$, we first map it onto the target manifold $\hat{y} = \hat{f}(x')$ and then classify $\hat{y}$ according to Equation (4.20). For evaluation, we use the classic Newsgroups-20[1] benchmark for hierarchical classification which comprises of 11314 training points, 7505 test points, and 20 classes. The augmented tree $\mathcal{T}_x$ consists of 11342 nodes and 36273 edges. We embed $\mathcal{T}_x$ in $\mathcal{L}^{10}$, what achieves an embedding quality of mAP 0.99 and mean rank 1.01. We then train HSP, NN-G and NN-E as described above and measure classification performance in terms of μF1 and macroF1 scores. We follow prior work and measure F1 scores over the leaf classes of the hierarchy.
Table 4.1 shows the results of our experiments. It can be seen that the hyperbolic structured predictor achieves results comparable to state-of-the-art on this task although we did not explicitly optimize the embedding or training loss for hierarchical classification. Moreover, it can be seen that geodesic NN-G clearly outperforms the two algorithms using an Euclidean loss: NN-E and KRLS.

Results Table 4.1 shows the results of our experiments. It can be seen that the hyperbolic structured predictor achieves results comparable to state-of-the-art on this task although

[1] http://qwone.com/~jason/20Newsgroups/

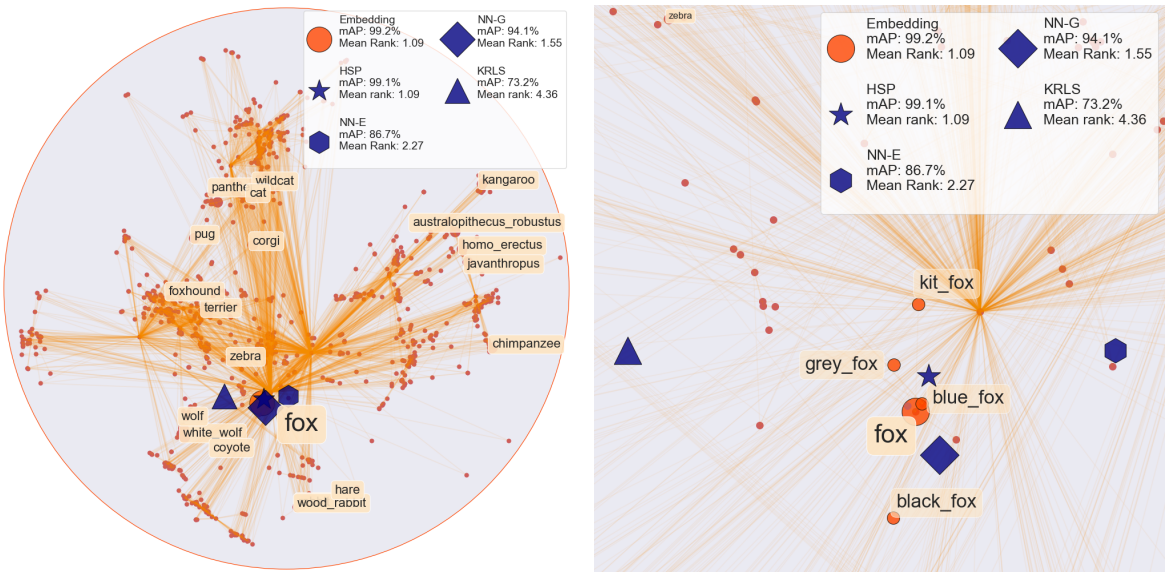
Table 4.1 Hierarchical classification on benchmark datasets. We report micro-F1 (μF1), macro-F1 (MF1), as well as the rank relative to all other models on a dataset, e.g., (1) for the the best performing model.

		Model - Performance (Relative Rank)									
		TD-LR		HSVM		NN-E		NN-G		HSP	
News-20	μF1	77.07	(3)	**80.79**	(1)	63.91	(5)	72.67	(4)	80.28	(2)
	MF1	77.94	(3)	**80.04**	(1)	64.21	(5)	72.70	(4)	79.56	(2)
Imclef07a	μF1	73.86	(3)	74.98	(2)	65.49	(5)	67.49	(4)	**75.95**	(1)
	MF1	36.03	(3)	**50.44**	(1)	26.76	(5)	31.20	(4)	46.41	(2)
Wipo	μF1	36.85	(2)	**38.48**	(1)	16.87	(5)	16.69	(6)	31.94	(3)
	MF1	52.18	(3)	52.21	(2)	42.77	(5)	42.86	(4)	**52.41**	(1)
Diatoms	μF1	**54.01**	(1)	48.97	(3)	9.25	(5)	11.31	(4)	53.20	(2)
	MF1	55.53	(2)	44.61	(3)	14.90	(4)	14.61	(5)	**62.10**	(1)
	Avg. Rank		(2.5)		(1.75)		(4.88)		(4.38)		(1.75)

we did not explicitly optimize the embedding or training loss for hierarchical classification. We also observe that while NN-G outperforms NN-E, both algorithms perform significantly worse on Wipo and Diatoms datasets. Interestingly, these two datasets are significantly smaller compared to Newsgroup-20 and Imclef07a in terms of number of training points ($\sim 1K$ Vs $\sim 10K$ training samples). This seems to suggest that NN-G and NN-E models have a higher sample complexity.

4.5.3 Taxonomy expansion

For taxonomy expansion, we assume a similar setting as for hierarchical classification. We are given a dataset $\mathcal{D} = \{x_i, c_i\}_{i=1}^m$ where concepts c_i are organized in a taxonomy $\mathcal{T}$ and for each concept we have an additional feature representation x_i. Again, we first embed the hierarchy $\mathcal{T}_x$ and split it into train $\mathcal{D}^e_{\text{train}}$ and test set $\mathcal{D}^e_{\text{test}}$. We vary the size of the test set (the number of unknown concepts) such that $|\mathcal{D}_{\text{test}}| \in \{5, 10, 20, 30, 50\}$. Whenever necessary, we also create a validation set from $\mathcal{D}_{\text{train}}$ for model selection with a 80 : 20 ratio for model selection. We then train all regression functions $\hat{f} : \mathbb{R}^d \to \mathcal{L}^n$ using $\mathcal{D}^e_{\text{train}}$ and predict embeddings for $\mathcal{D}_{\text{test}}$. In contrast to hierarchical classification, the predicted points $\hat{y} = f(x)$ can themselves act as parents of other points, i.e., they are full members of the taxonomy $\mathcal{T}$. To assess the quality of the predictions we use mean average prediction (mAP) as proposed by Nickel and Kiela (2017). We report mAP for the predicted points as well as for the points originally embedded by the Lorentz embedding (Orig). This



(a) WordNet mammals embedding

(b) Close-up of predicted embedding for 'Fox'

Figure 4.4 Overview and close-up of predicted positions for entity 'Fox'. Models that do not use the geometry of the hyperbolic manifold fail at positioning the entity, while the geodesic neural network and the hyperbolic structured predictor position the entity accordingly to its real neighbours. Only the first 2 dimensions of a 5-dimensional embedding are represented.

experiment is repeated 20 times for a given size of the test set, each time selecting a new training-test split. In our experiments, we consider the following datasets:

WordNet Mammals. For WordNet Mammals, the goal is to expand an existing taxonomy by predicting concept embeddings from text. For this purpose, we take the mammals hierarchy of WordNet and retrieve for each node its corresponding Wikipedia page. If a page is missing, we remove the corresponding node and if a page has multiple candidates we disambiguate manually. The transitive closure of $\mathcal{T}$ has 1036 nodes and 11222 edges. Next, we pre-process the retrieved Wikipedia descriptions by removing all non alphabetical characters, tokenizing words and removing stopwords using NLTK (Loper and Bird, 2002). Finally, we associate to each concept $c_i \in \mathcal{T}$ the TF-IDF vector of its Wikipedia description as feature representation $x_i \in \mathbb{R}^{10000}$ computed using Scikit-learn (Pedregosa et al., 2011). In the experiments we noticed that removing the taxonomy names from the TF-IDF features does not affect significantly the outcome of the experiments. We then embed $\mathcal{T}$ following Section 4.2 and obtain an embedding with mAP 0.86. This dataset is particularly difficult given the way features were collected: Wikipedia pages have a high variance in

Table 4.2 Mean average precision for taxonomy expansion on WordNet mammals and synthetic data

		Number of new concepts				
		5	10	20	30	50
Wordnet Mammals	Orig	0.86 ± 0.06	0.88 ± 0.06	0.87 ± 0.03	0.87 ± 0.03	0.88 ± 0.02
	KRLS	0.54 ± 0.14	0.37 ± 0.07	0.26 ± 0.04	0.22 ± 0.03	0.15 ± 0.02
	NN-E	0.61 ± 0.12	0.47 ± 0.08	0.38 ± 0.04	0.31 ± 0.03	0.20 ± 0.03
	NN-G	0.79 ± 0.08	$\mathbf{0.74 \pm 0.06}$	0.63 ± 0.06	$\mathbf{0.61 \pm 0.06}$	$\mathbf{0.50 \pm 0.04}$
	HNN	$\mathbf{0.82 \pm 0.05}$	0.73 ± 0.05	0.63 ± 0.04	0.57 ± 0.05	0.46 ± 0.04
	HSP	0.72 ± 0.10	0.69 ± 0.07	$\mathbf{0.69 \pm 0.07}$	0.58 ± 0.09	$\mathbf{0.50 \pm 0.06}$
Synthetic Small	Orig	0.94 ± 0.03	0.93 ± 0.03	0.94 ± 0.02	0.94 ± 0.02	0.94 ± 0.01
	KRLS	0.63 ± 0.16	0.51 ± 0.12	0.36 ± 0.06	0.27 ± 0.03	0.21 ± 0.02
	NN-E	0.76 ± 0.07	0.72 ± 0.09	0.63 ± 0.09	0.56 ± 0.09	0.45 ± 0.08
	NN-G	0.80 ± 0.07	0.73 ± 0.06	0.61 ± 0.06	0.55 ± 0.05	0.45 ± 0.04
	HNN	$\mathbf{0.82 \pm 0.01}$	0.71 ± 0.07	0.60 ± 0.05	0.51 ± 0.04	0.41 ± 0.04
	HSP	$\mathbf{0.82 \pm 0.08}$	$\mathbf{0.76 \pm 0.07}$	$\mathbf{0.66 \pm 0.05}$	$\mathbf{0.60 \pm 0.04}$	$\mathbf{0.50 \pm 0.03}$
Synthetic Large	Orig	0.81 ± 0.06	0.79 ± 0.05	0.80 ± 0.03	0.80 ± 0.02	0.80 ± 0.01
	KRLS	0.30 ± 0.05	0.20 ± 0.02	0.13 ± 0.01	0.09 ± 0.01	0.07 ± 0.00
	NN-E	0.69 ± 0.09	0.68 ± 0.09	0.64 ± 0.05	0.61 ± 0.05	0.59 ± 0.05
	NN-G	0.77 ± 0.07	0.72 ± 0.07	0.71 ± 0.04	$\mathbf{0.69 \pm 0.04}$	$\mathbf{0.65 \pm 0.03}$
	HNN	$\mathbf{0.83 \pm 0.7}$	$\mathbf{0.79 \pm 0.4}$	$\mathbf{0.72 \pm 0.06}$	0.64 ± 0.06	0.63 ± 0.02
	HSP	0.76 ± 0.09	0.70 ± 0.07	0.69 ± 0.04	0.67 ± 0.05	0.63 ± 0.03

quality and amount of content, while some pages are detailed and rich in information other barely contain a full sentence.

Synthetic datasets. To better control for noise in the feature representations, we also generate datasets based on synthetic random trees, i.e., a smaller tree with 226 nodes and 1228 edges and a larger tree with 2455 nodes and 30829 edges after transitive closure. For each node we take as feature vector the corresponding row of the adjacency matrix of the transitive closure of the tree. We project these rows on the first d principal components of the adjacency matrix, where $d = 50$ for the small tree and $d = 500$ for the big tree. We then embed the nodes of the graph in $\mathcal{L}^5$ using both the tree structure and similarity scores computed using the vector features. The similarity is computed by a Gaussian kernel with σ equal to the average tenth nearest neighbour of the dataset.

Results We provide the results of our evaluation for different sizes on $\mathcal{D}^e_{\text{test}}$ in table 4.2. It can be seen that all hyperbolic-based methods can successfully predict the embeddings of unknown concepts when the test set is small. The performance degrades as the size of the test set increases, since it becomes harder to leverage the original structure of the graph. While all methods are affected by this trend, we note that algorithms using the geodesic loss tend to perform better than those working in the linear space. This suggest that taking into account the local geometry of the embedding is indeed beneficial in estimating the relative position of novel points in the space.

We conclude by noting that all hyperbolic-based methods have comparable performance across the three settings. However, we point out that HSP and NN-G offer significant practical advantages over HNN: in all our experiments they were faster to train and in general more amenable do model design. In particular, since HSP is based on a kernel method, it has relatively fewer hyperparameters and requires only solving a linear system at training time. NN-G consists of a standard neural architecture with the homeomorphism activation function introduced in section 4.4.2 and trained with the geodesic loss. This allows one to leverage all current packages available to train neural networks, significantly reducing both modeling and training times.

Part III

Structured Machine Learning Robotics

Chapter 5

Inverse Kinematics with Structured Prediction

Given the mechanical model of a robot, the configuration space is defined as the set of all possible positions that the robot joints can attain. The workspace is the set of all possible positions and orientations that the end effector of the robot can reach. Computing the kinematics of a robot consists in finding a function that maps the configuration space to the workspace. In many practical situations such as robotic control or kinematics calibration, it is often more interesting to compute the inverse of this function (Spong et al., 2006), which is reffered to as the inverse kinematics problem.
Computing the inverse kinematics of a robot is a task traditionally solved by analyzing the geometric model of the robot. A major problem with this approach is that the solution to this problem is only as good as the model of the robot. A recently proposed alternative is to learn the inverse kinematics functions from sampled pairs of configuration-workspace points (De Angulo and Torras, 2008; D'Souza et al., 2001; Oyama et al., 2001, 2005). However, traditional regression techniques are not suited for this task. For robots with redundant joints inverse kinematics is an ill-posed problem since there are multiple joint configurations that correspond to the same workspace point (Siciliano, 1990). To overcome this issue, various works have reframed the problem in the velocity domain. The goal becomes learning a map from the velocity of the end effector to the velocity of joints. Because this problem is locally linear (Tevatia and Schaal, 2000), regression techniques can be employed to compute piecewise functions. This approach has been explored both with linear functions and NNs (Oyama et al., 2001; Tevatia and Schaal, 2000). Another regression-based approach was proposed by Jordan and Rumelhart (1992), where a NN learns the forward kinematics and is used to train a second NN such that the composition

of the two is the identity function. However the non-convex nature of miminization problems generated by NN models led to instability during training. Bócsi et al. (2011) proposed an approach based on structured predictors. By training a one-class structured SVM they were able to learn the inverse kinematics for some trajectories; however this came at the cost of sampling the training set only in a neighbourhood of the goal trajectory. This poses a problem as it requires to retrain the model for each new trajectory or limit the usage of a model to trajectories close to each other.
In this chapter we follow the line of work of Bócsi et al. (2011), improving it. We learn the inverse kinematics of a robot with the goal of finding the joint configurations for a trajectory in the workspace. However we test our approach on a more challenging task. We solve the inverse kinematics also for the end effector orientation in addition to its position; this problem is called pose estimation. We show that it is possible to do so using only a limited amount of samples collected from the whole workspace of the robot. We do so by proposing a structured prediction algorithm and comparing it with an unstructured multivariate neural network and a one-class structured SVM. We show the limits of both the one class structured SVM and the unstructured approach when dealing with such problem and how our approach leads to higher accuracy and reduced training times.
The remainder of the chapter is organized in the following way. In Section 5.1 we introduce the problem of inverse kinematics, recall the one class SVM used by Bócsi et al. (2011) and the neural network used as unstructured baseline. Then we go over our proposed algorithm for the task: a consistent regularized structured predictor as introduced in Chapter 2. In Section 5.1.4 we introduce the task of trajectory reconstruction; then in Section 5.2 we compare the algorithms on a test set and 2 trajectories of a simulated robotic arm.

5.1 Problem and proposed approach

Consider the space of end effector position and orientation $\mathcal{X} = \mathbb{R}^d \times [0, 2\pi)$ and the space of joint configurations $\mathcal{Y} = [y_{1,1}, y_{1,2}] \times [y_{2,1}, y_{2,2}] \times \ldots \times [y_{J,1}, y_{J,2}]$ where $y_{j,1} < y_{j,2}$ and $y_{j,i} \in [0, 2\pi)\ \forall i \in \{1, 2\},\ \forall j = 1, \ldots, J$. Here each couple $[y_{j,i}, y_{j,i}]$ represents the joint boundaries of the J-th joint. The goal is to find a function $g^{-1} \colon \mathcal{X} \to \mathcal{Y}$ such that

$$g^{-1}(x) = y \tag{5.1}$$

is compliant with the geometry of the robot mapping. However finding such function is not straightforward, since the direct kinematics function is, in general, surjective. We propose

a data-driven solution to this problem using structured machine learning techhniques. Given a sample of joint configurations and corresponding end effector pose $\{x_i, y_i\}_{i=1}^n$, we try to *learn* a function $\widehat{f}(x) \approx g^{-1}$ by minimizing an error measure. We compare three different machine learning algorithms on this task. A NN trained to perform multivariate regression onto the joint space, a one class structured SVM as proposed by Bócsi et al. (2011), and a consistent regularized structured predictor as introduced in Section 2.2. Only the last two algorithms account for the structure in the output space.

5.1.1 One-Class Strucured SVM

To learn the inverse kinematics of redundant robots, Bócsi et al. (2011) proposed to use one-class SVM (OCSVM). This algorithm is a type of SSVM originally employed for density estimation tasks. Minimization problems using SSVM are not solvable for dense output spaces, and this limit is overcome by OCSVM which are used to estimate a continuous distribution $\hat{\rho}(y|x)$. Given a joint kernel $k\colon (\mathcal{X} \times \mathcal{Y}) \times (\mathcal{X} \times \mathcal{Y}) \to \mathbb{R}$, OCSVM learns the following prediction function

$$\widehat{f}_{OC}(x) = \arg\max_{y \in \mathcal{Y}} \sum_{i=1}^{n} \alpha_i k\big((x, y), (x_i, y_i)\big) \tag{5.2}$$

where $a_i \in \mathbb{R}\ \forall\ i = 1, \ldots, n$. The weights a_i are determined by solving

$$\min_{a \in R^n} \sum_{i}^{n} \sum_{j}^{n} a_i a_j k\big((x_i, y_i), (x_i, y_j)\big) - \sum_{i=1}^{n} a_i k\big((x_i, y_i), (x_i, y_i)\big) \tag{5.3}$$

subject to, $\forall\, i = 1, \ldots, n$

$$0 \leq a_i \leq \frac{1}{\lambda n} \qquad\qquad \sum_{i=1}^{n} a_i = 1 \tag{5.4}$$

with $\lambda \in (0, 1]$ a regularization parameter to be chosen by the user. Notice how this problem is akin to the one defined for training kernelized SSVM in Equation (2.8). For a derivation of this algorithm we refer the reader to the work of Schölkopf et al. (2001). To solve Equation (5.3) we employ the path-following algorithm (Vandenberghe, 2010) implemented in the *CVXOPT* python package (Andersen et al., 2013). To evaluate Equation (5.2) we use the Broyden-Fletcher-Goldfarb-Shanno algorithm (BFGS) algorithm, a gradient-based quasi-Newton method for unconstrained minimization (Nocedal and Wright, 2006).

5.1.2 Multivariate Neural Network Regression

As a baseline we propose to model the inverse kinematics function with an unstructured neural network (NN) $\widehat{f}_{NN}\colon \mathcal{X} \to \mathbb{R}^J$ with 5 layers. We choose a NN over other models for the recent outstanding results this family of models has obtained in a variety of regression and classification tasks (Lathuilière et al., 2019). We define each layer to have respectively $\{16, 64, 64, 32, J\}$ hidden units, where J is the number of joints in the robot. We use hyperbolic tangent as non-linearity in every layer except the output one, where no non-linearity is used. We did not notice any increase in performance by either adding hidden units or layers to the model or changing activation function. Because the NN is agnostic to the structure of $\mathcal{Y}$ its actual domain is $\mathbb{R}^J$ which is a superset of $\mathcal{Y}$. To avoid predicting invalid configurations, we project any prediction $\widehat{f}_{NN}(x) \in \mathbb{R}^J$ onto the actual output space $\mathcal{Y}$ by thresholding element-wise all the values outside of the corresponding joint angle range.

5.1.3 Consistent Resgularized Structured Predictor

As a structured alternative to OCSVM we propose a structured predictor that stems from the CRiSP framework detailed in Section 2.2. We argue that this method requires significantly less samples than OCSVM to be trained. In the CRiSP framework the predictor is defined as

$$\widehat{f}_{CR}(x) = \operatorname*{arg\,min}_{y \in \mathcal{Y}} \sum_{i=1}^{n} \alpha_i \Delta(y_i, y) \tag{5.5}$$

where $\alpha(x) = [\alpha_1, \ldots, \alpha_n]^\top = \left(K + n\lambda I_n\right)^{-1} K_x$, K is the kernel matrix of a fixed kernel $k\colon \mathcal{X} \times \mathcal{X} \to \mathbb{R}$, with entries $\{K\}_{ij} = k(x_i, x_j)$, $K_x \in \mathbb{R}$ is the vector with entries $\{K_X\}_i = k(x, x_i)$ and $I_n \in \mathbb{R}^{n \times n}$ is the identity matrix. Since the output space is a subset of Euclidean space, we choose the loss to be the squared distance $\Delta(y_1, y_2) = \|y_1 - y_2\|^2$, and we account for the structure of the output domain by using the BFGS method for constrained functions (Byrd et al., 1995) to minimize Equation (5.5), where the constraints are the ones defined by the joint limits. This ensures that any resulting $\widehat{f}_{CR}(x)$ is within the boundaries of $\mathcal{Y}$ for any $x \in \mathcal{X}$.

5.1.4 Trajectory reconstruction

In many robotics applications it is necessary to solve the inverse kinematics for a sequence of points $\{x_t\}_{t=1}^{L}$ which describes a trajectory in space. Bócsi et al. (2011) propose to compute the inverse kinematics of a trajectory one point at a time, using the inferred $\widehat{f}(x_{t-1})$ of the previous trajectory point, as the initial value for the minimization problems

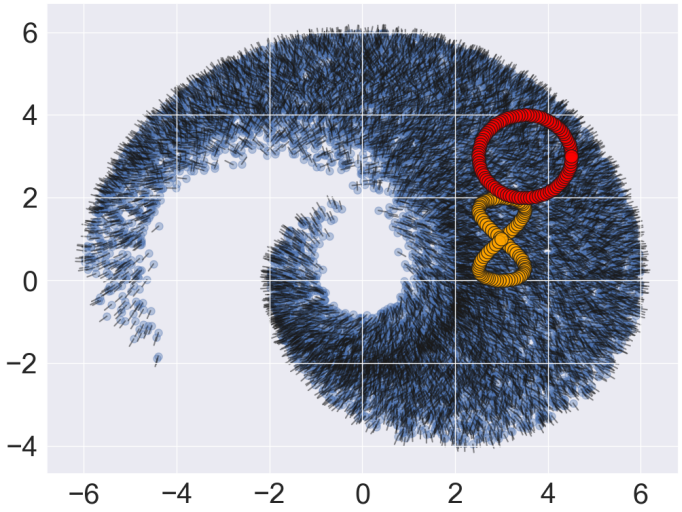


Figure 5.1 A plot showing the sampled workspace as blue markers for positions, black short line for orientation, and the two trajectories used in the trajectory reconstruction task. The fixed base of the robot is at $(0,0)$.

of Equation (5.2) and Equation (5.5). This idea hinges on the assumption that close points in workspace have close solutions in configuration space, thus employing gradient based techniques to evaluate the estimator will lead to such solution.

5.2 Experimental validation

We assess the performance of the proposed methods on a simulated planar robotic arm with 3 degrees of freedom. We sample 20000 points uniformly at random in the configuration space $\mathcal{Y}$ and generate the corresponding points in the workspace $\mathcal{X}$, we use 14000 of these points as training set (shown in Figure 5.1), 3000 as validation set and 3000 as test set. We cross-validate the OCSVM and the CRiSP hyperparameters on the validation set with grid search. For the OCSVM we choose the kernel

$$k_{OC}\big((x_1,y_1),(x_2,y_2)\big) = e^{-g\|\bar{\psi}_{OC}(x_1,y_1)-\bar{\psi}_{OC}(x_2,y_2)\|^2} \tag{5.6}$$

with $\bar{\psi}_{OC} = \big[x,\ cos(x),\ sin(x),\ sin(x)cos(x),\ y,\ sin(y),\ cos(y),\ sin(y)cos(y)\big]^\top$ and $g \in \mathbb{R}$ the kernel bandwidth. Here $sin(\cdot)$ and $cos(\cdot)$ are the corresponding element-wise trigonometric functions. We choose the respresentation $\bar{\psi}_{OC}$ noticing an increase in

Table 5.1 Mean Squared Error (MSE) scaled by a factor of $1e2$ and Explained Variance (AE) for test set and trajectory reconstruction.

		NN	CRiSP	OCSVM
Test	MSE	3.8 ± 6.4	$\mathbf{1.6 \pm 4.4}$	5.1 ± 39
	EV	0.984	**0.993**	0.846
Eight	MSE	0.042 ± 0.22	$\mathbf{0.018 \pm 0.017}$	56 ± 23
	EV	0.958	**0.998**	0.80
Circle	MSE	49.35 ± 47.39	$\mathbf{0.173 \pm 0.137}$	70 ± 29
	EV	0.67	**0.998**	0.60

performance during cross-validation when compared to the one chosen by Bócsi et al. (2011). For the CRiSP we use the following kernel

$$k_{CR}(x_1, x_2) = e^{-g\|\bar{\psi}_{CR}(x_1) - \bar{\psi}_{CR}(x_2)\|^2} \tag{5.7}$$

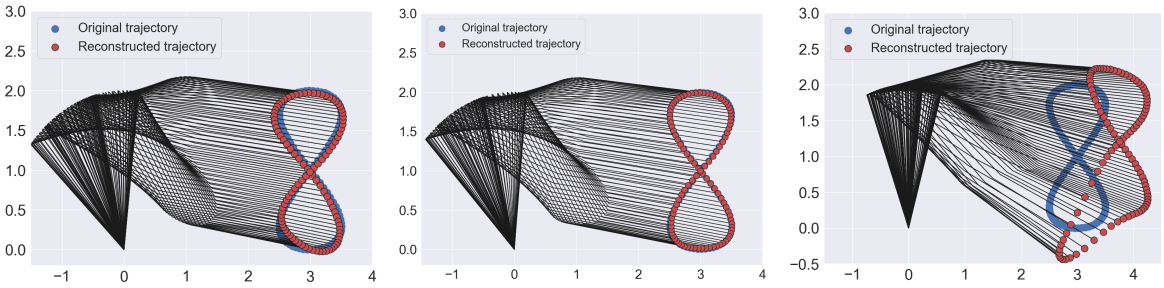
with $\bar{\psi}_{CR}(x) = \left[x,\ sin(x),\ cos(x),\ sin(x)cos(x),\ sin(x)^2,\ cos(x)^2\right]^\top$. We train the NN using Adam optimizer (Kingma and Ba, 2014) and early stopping on the validation set error. The training tipically converges after 2000 iterations. We test the trained models on two tasks: error on the test set and trajectory reconstruction.

Test set For each point in the test set we predict the corresponding joints configuration and compare it with the true value by using mean squared error, we also computed the explained variance over all the test points. The test set is sampled uniformly at random from the workspace of the robot.

Trajectory reconstruction We create two trajectories in workspace of 100 points each, and using the trajectory reconstruction approach by Bócsi et al. (2011) we predict the corresponding joint configuration $\widehat{f}(x) = \hat{y}$. We then use $\hat{y}$ to compute the actual pose of the end effector and measure the mean square error with respect to the desired trajectory point and the explained variance. We generate 2 trajectories, both depicted in Figure 5.1. The first is an eight-shaped trajectory within the workspace of the robot. The second is a circle for which we impose an orientation such that the points farthest from the origin are barely unreachable with such orientation. We do so in order to evaluate the behaviour of the compared methods in the case of an unreachable pose.

Table 5.2 Experimental training time on 14000 points for NN, CRiSP and OCSVM over 12 repetitions

	Training time in minutes
NN	$5'8" \pm 1"$
CRiSP	$1'50" \pm 1'45"$
OCSVM	$12'38" \pm 5'43"$

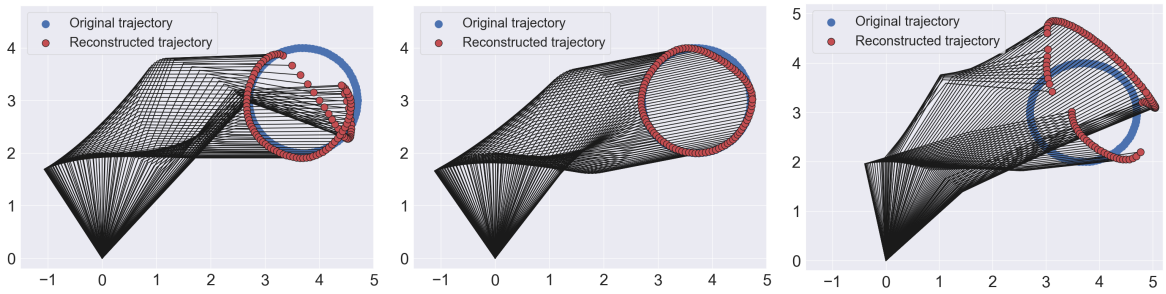


(a) NN (b) CRiSP (c) OCSVM

Figure 5.2 Trajectory reconstruction on eight-shaped trajectory. (a) NN. (b) CRiSP. (c) OCSVM.

5.2.1 Results

Experimental results are reported in Table 5.1. It can be observed that the CRiSP is the algorithm performing best in every task, followed by the NN which, altough unstructured, outperforms the OCSVM. This can be observed qualitatively in Figures 5.2a to 5.2c. We argue that this is due to intrinsic limitations of OCSVM, which was originally developed for density estimation purposes. In the original work by Bócsi et al. (2011) the OCSVM was trained with points sampled close to the test trajectory, while we randomly select points from the whole workspace. Moreover we use a number of training samples less than a half of the ones used by Bócsi et al. (2011). Looking at Figure 5.3 it is possible to observe the behaviour of the three algorithms when presented with points that does not belong to the workspace. The OCSVM shows an erratic trajectory. The NN predictions result either in accurate predictions or non-valid joint configurations, 29 of the NN predictions for the circle trajectory resulted in non valid joint configurations that were projected to configuration space. On the other hand, the CRiSP shows a smoother approximation of the points that cannot be reached by the robot, with a smaller loss. We also show empirical training times in Table 5.2, showing how training a CRiSP is significantly faster.



(a) NN (b) CRiSP (c) OCSVM

Figure 5.3 Trajectory reconstruction on circumference trajectory. (a) NN. (b) CRiSP. (c) OCSVM.

Chapter 6

Faster Biological Motion Detection

The task of biological motion detection consists in discriminating whether a movement observed by a robot is either generated by a human or a non-biological entity such as a toy, a car, weather conditions, etc. In general, perceiving the world represents a core skill for autonomous agents that need to interact with the surrounding environment. While recognizing and localizing objects might be fundamental for a scene understanding purpose, the agents might be asked to accomplish more complex tasks which might need a deeper comprehension of the activities around them. Especially, for those agents which are required to interact with humans in order to help them or use tools (e.g. collaborative robots in work places or humanoids such as iCub or the R1 robotic platforms (Metta et al., 2010; Parmiggiani et al., 2017)), the ability to discriminate between movements generated by a collaborator or another non-biological system in the background, is essential. Evidences of the fundamental relevance of this skill can be found in human infants and their development. Specifically, although their sensory-motor capabilities are limited and typically they still need to learn standard social norms, they are naturally able to identify other humans in their surroundings. Indeed, they clearly manifest a preference for biological motion (Simion et al., 2008) and for faces looking directly at them (Farroni et al., 2002) which facilitate their development.
Moreover, the motivation of this work relies on its practical relevance. Numerous real world applications would benefit from the integration of a reliable biological motion detection algorithm since it represents the first module of a more task specific pipeline. For instance, in a surveillance system, the detection of a biological motion can trigger more sophisticated person identification algorithms. It is also necessary for underwater robots that need to track and interact with humans, discarding all the other moving elements of the surroundings (Sattar and Dudek, 2018). Furthermore, it can be used as initial step for

the cooperation with a human. A collaborative robot, indeed, might be asked to detect a working human, possibly partially or mostly hidden, among other moving non-biological systems.
In this work we address the task considering a sequence-to-sequence classification problem to discriminate biological from non-biological motion in video frames. By forgoing the binary classification approach originally proposed by Vignolo et al. (2017) we leverage the structure of the input and output domain and improve the original pipeline in terms of computational efficiency. We build on the original pipeline that makes use of a temporal multi-resolution motion feature which automatically copes with different dynamics and builds on top of low-level features that capture biological motion regularities. The starting point of the representation is the optical flow, a low-level measurement which simulates the limited amount of visual information available at birth.
Our contribution is threefold. First, we acquire a novel multi-view dataset representing indoor biological and non-biological actions, on which we carry out the evaluation of our method. The samples in the dataset are labeled in such a way that they can be used also for different tasks with respect to the one described in this document, such as action recognition, or multi-view motion analysis. Second, we build on the pipeline described by Vignolo et al. (2017) and introduce a different learning model, a Gated Recurrent Unit network, to better handle the intrinsic structure of the problem. Finally, we show how this different model allowed us to improve the performances of the original work.
The chapter is organized as follows: in Section 6.1 we describe the biological motion detection task, providing details about the two thirds power law. Then, in Section 6.2, 6.3 and 6.4 we illustrate the main contributions of this work, respectively the novel collected dataset, the proposed pipeline and the results obtained in our empirical evaluation.

6.1 Biological Motion Detection

The task of biological motion detection (BMD) can be described as identifying the source of movement in a sequence of frames and deciding if the movement is generated by a biological entity (such as a human) or non-biological entity (such as a machine or a toy). One of the first approaches proposes to use RGB and infrared information to discriminate entities based on face information in the thermal spectrum (Correa et al., 2012), however infrared cameras are expensive and not a standard in most robotics platform as opposed to simpler RGB-depth cameras (Koppula et al., 2013; Sung et al., 2012). Indeed, because BMD is typically a task supporting more complex behaviours (Bisio et al., 2014; Mutlu et al.,

Figure 6.1 Sample frames drawn from the proposed dataset: (a) a *folding* action, from a lateral point of view, (b) a *packing* action, from a frontal point of view, and (c) a *Toy Train* sequence, from a frontal point of view.

2012; Sciutti et al., 2012), hardware constraints become even more important. Vignolo et al. (2017) introduced a method to detect biological movement by using only RGB frames, however this method introduces a latency of 30 frames in the processing pipeline and uses a kernel-based classifier. Because kernel classifiers use the training set to compute a classifying function, the memory and computation efficiency get worse as the dataset size increases. With this work we propose an improvement on this approach that has fixed memory and computation cost while also reducing the 30 frames delay to a 5 frames delay. In the following we recall the mathematical model for characterizing biological movements introduced by Noceti et al. (2015), as it is the basis for the biological motion detection application discussed in this chapter.

6.1.1 Characterization of biological movement

Various works have shown that humans can easily recognize and predict movements that follow human-based kinematics (Gavazzi et al., 2013; Pozzo et al., 2006; Viviani et al., 1997). While these motions are often the result of complex dynamics and subtle visual cues, humans can easily detect and interpret them based solely on a limited number of landmarks on the moving entity (Blakemore and Decety, 2001). This suggests that there are simple ways of characterizing biologically generated motions. For this purpose Noceti et al. (2015) introduced a set of time-varying descriptors of human movements, based on the *two-*

thirds power law (Lacquaniti et al., 1983). These descriptors are velocity (Equation (6.1)), acceleration (Equation (6.2)), curvature (Equation (6.3)) and radius of curvature (Equation (6.4)):

$$V_t = [u_t, v_t, \Delta_t] \in \mathbb{R}^3 \tag{6.1}$$

$$\|C_t\| = \frac{\|V_t\| \times A_t}{\|V_t\|^3} \tag{6.2}$$

$$A_t = [u_t - u_{t-1}, v_t - v_{t-1}, 0] \in \mathbb{R}^3 \tag{6.3}$$

$$\|R_t\| = \frac{1}{\|C_t\|} \tag{6.4}$$

where (u_t, v_t) is the vector representing the instantaneous velocity along vertical and horizontal axis, t is the time index and Δ_t is the sampling interval (for video-based applications the sampling interval corresponds to the frames-per-second of the camera). By processing subsequent frames it is possible to compute these descriptors locally for each pixel and obtain information on movements happening in various regions of the scene, our goal is to use this information to determine the nature of the movement.

6.2 Dataset

In order to evaluate the quality of the proposed approach we acquired a dataset of human-generated and object-generated movements from a robot point of view. While in our work the primary task is biological motion detection, the dataset structure allows for more general applications such as action recognition or person classification.
The BMD Dataset is a set of videos recorded from a Intel® RealSense™ D435 RGB camera, mounted on the R1 robot (Parmiggiani et al., 2017), each video lasts approximately 20 seconds and comprises of a single biological or non-biological motion or a mix of both types. All videos are acquired at 15 frames per second with 640×480 resolution and encoded using H.264 codec. Figure 6.1 shows some sample frames from three sequences drawn from our dataset.

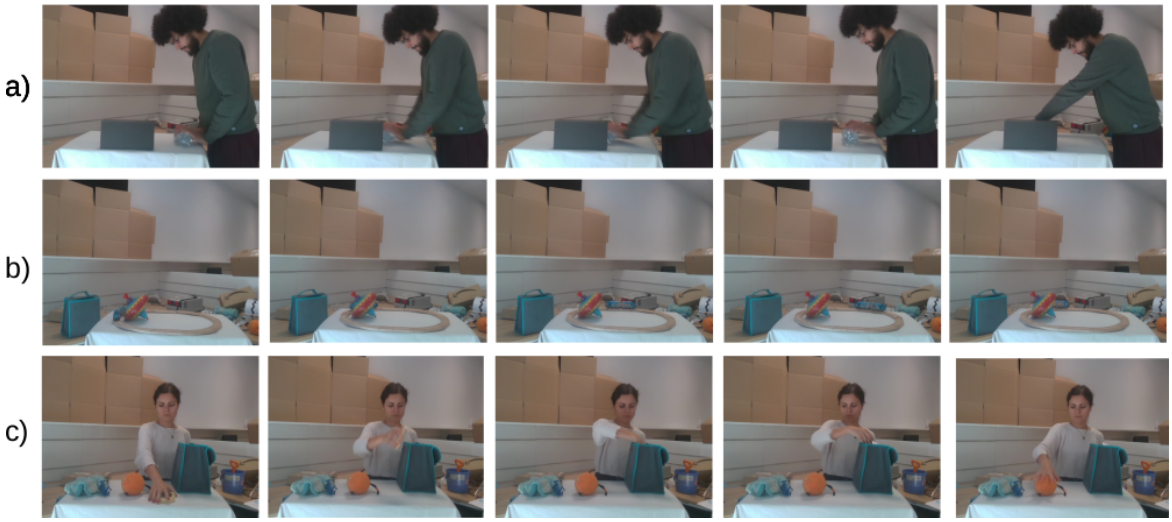


Figure 6.2 Sample frames from test set videos: (a) *Rolling* action partially occluded by a box, (b) *Toy-Train* with tracks partially covered by a toy top, (c) new subject performing the *Packing* action, not present in the training set.

6.2.1 Training set

Biological motions Biological motion videos are grouped based on the type of action, the subject performing the action and the point of view of the robot with respect to the subject. Each action-subject-view configuration is recorded 2 times. In the dataset there are a total of 6 actions: *Drawing* on a piece of paper, *Folding* some cloths, *Mixing* liquids in a bucket, *Transporting* a small object on a table, *Pointing* at different locations in space, *Rolling* a bottle full of water (which emulates the dynamics of rolling cooking dough). Each action is performed repeatedly until roughly 20 seconds are recorded. Each action is performed by 3 subjects, one per video, from 2 points of view. The primary view has the robot standing in front of the subject, the secondary view is taken from roughly 45° from the front of the robot. In total there are 72 biological motion videos in the training dataset.

Non-biological motions Non-biological videos are grouped based on the object generating the motion and the point of view of the robot. Each movement-view configuration is recorded 4 times. The non-biological motions are 7: a *Balloon* moved by an electric fan, a display playing a video of *Clouds* moving, a cylinder presenting a *Random Pattern* rotating, a cylinder showing a *Regular Pattern* rotating, a *Toy Top* rotating, a *Toy-Train* running on elliptic tracks and a *Empty* scene with no movement. We record 4 views for each object: primary close, primary farther, secondary close and secondary farther. Primary and secondary refer to the robot standing respectively in front and 45 degrees from the

front with respect to the motion, while close and farther correspond to a distance of 1 meter and 2 meters from the object moving. In total there are 112 non-biological motion videos in the training dataset.

Mixed videos Because in this chapter we propose an approach based on sequence classification, we also gather videos where the source of the motion changes from biological to non-biological or vice-versa. Since transitions can lead to noisy visual effects we want to test the capability of our approach to deal with these events. We mix two of the previously described movements in a single video, each video starts with a non-biological motion, then a human subject enters the scene stopping the non-biological source of movement and starts performing one of the aforementioned actions. We also record the inverse: a subject performing an action, followed by a non-biological movement, without any other source of movement. The biological/non-biological pairs of motions are *Mixing*/*Toy-Train*, *Moving*/*Toy-Train*, *Pointing*/*Random Pattern* and their inverses. Each video lasts 20 seconds and is taken from one of two possible point of view 3 times. This repeated for each subject for a total of 108 videos.

6.2.2 Test set

In a similar spirit to training set, we acquire a test set and use it to evaluate the quality of our approach in different conditions. For biological motions we record a new subject performing the same actions present in the training set plus 2 novel actions: *Waving* and *Packing* items in a bag. All videos are recorded from two point of views as in the training set. Then we record videos of one of the subjects present in the training set, performing a subset of the original actions while the view is partially *occluded*. The action are: *Folding*, *Moving* an object, *Pointing*, *Rolling*. We also record 2 views of the new actions (*Waving* and *Packing*) performed by one of the subject already present in the training set. For non-biological motion we acquire the video of an *Electric Fan* rotating and a *Toy-Train* running on tracks with part of the path hidden from the camera. Both actions are recorded with and without a human present in the video. Example frames from the videos are shown in Figure 6.2.

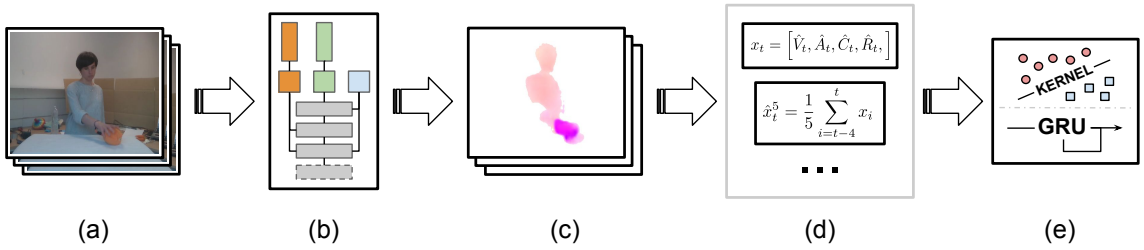


Figure 6.3 A general overview of the pipeline of our method: (a) given an input sequence, (b-c) we estimate the optical flow by means of PWC-Net (Sun et al., 2018), (d) we compute a set of biological motion descriptors which we finally feed to (e) the chosen learning algorithm.

6.3 Accelerated pipeline

Figure 6.3 gives an overview of the pipeline of our method. Given a sequence of images, of length t, our system produces a binary answer associated with the biological motion classification problem. First, for every frame in the input video, we extract the visual features corresponding to the motion taking place in the scene, as described in Vignolo et al. Vignolo et al. (2017), by means of the chosen optical flow extraction algorithm. We then carry out temporal smoothing on the resulting features to stabilize their behavior over time. Finally, the smoothed features are fed to a classification model, which we have previously trained on the dataset described in Section II.

6.3.1 Visual Features Extraction

In order to characterize mathematically the movement taking place in a video we compute the optical flow of its stream of frames, as done by Vignolo et al. (2017). More precisely, given a pair of images $\mathbf{I}_0$ and $\mathbf{I}_1$, an optical flow estimation algorithm produces a 2D vector field $(u, v)\colon \mathbf{I} \to \mathbb{R}^2$ such that, for every pixel $p \in \mathbf{I}_0$, $(u(p), v(p))$ represents its instantaneous velocity (i.e. the estimated translation it underwent between frame $\mathbf{I}_0$ and frame $\mathbf{I}_1$). This quantity is the basic building block for the descriptors introduced in Section 6.1, allowing us to compute velocity, acceleration, curvature and radius of curvature for the whole image.

Our optical flow estimation method of choice is PWC-Net, a deep convolutional architecture introduced by Sun et al. (2018). Our selection has been driven both by computational requirements (we aim at real-time performance) and by the accuracy in the estimation process, since a more precise estimate leads to better movement descriptors.

Because optical flow algorithms are prone to generating noisy outputs even over pixels with no actual movement, we suppress the magnitude of velocity whenever this is under some fixed small threshold. We group the remaining points in connected components, of which we only consider the biggest one, that we denote with $\mathcal{R}$. Empirically we notice that this choices allows us to correctly select the source of the movement, independently of its nature. This procedure leads to a vectorial representation of an image that has non-fixed size, depending on the amount of pixels moving between each frame. To overcome this problem we average the norm of the four descriptors across the grouped pixels obtaining a 4-dimensional representation $x_t \in \mathbb{R}^4$ for each frame:

$$\hat{V}_t = \frac{1}{N}\sum_{p\in\mathcal{R}} \left\| V_t(p) \right\|, \qquad \hat{A}_t = \frac{1}{N}\sum_{p\in\mathcal{R}} \left\| A_t(p) \right\|,$$

$$\hat{C}_t = \frac{1}{N}\sum_{p\in\mathcal{R}} \left\| C_t(p) \right\|, \qquad \hat{R}_t = \frac{1}{N}\sum_{p\in\mathcal{R}} \left\| R_t(p) \right\|$$

$$x_t = \left[\hat{V}_t, \hat{A}_t, \hat{C}_t, \hat{R}_t, \right] \tag{6.5}$$

where N is the number of points belonging to $\mathcal{R}$. The sequence of vectors $\{x_t\}_{t=1}^{T}$ encapsulates the information describing the behaviour of the main moving entity in the video. However, even after averaging across $\mathcal{R}$, the resulting representation is characterized by very high frequency changes over time, which affected negatively the results achievable within our pipeline. To obtain a more stable signal, in the spirit of Vignolo et al. (2017), we compute the moving average over the past 5 frames: $\hat{x}_t^5 = \frac{1}{5}\sum_{i=t-4}^{t} x_i$. To allow for online robotics applications we consider only past frames. This step effectively introduces a 5 frame delay, as opposed to the 30 frames delay introduced by the longer filters in Vignolo et al. (2017). By concatenating x_t and $\hat{x}_t^5$ we obtain a 8-dimensional representation of each frame.

6.3.2 Sequence classification for biological motion detection

Given a set of vectorial features as described in Section 6.3.1 it is possible to classify each frame as containing either a biological or non-biological movement. Given the generality of the representation it is possible to apply a variety of algorithms to perform this task. We consider two approaches. The first is the one proposed in the original work by Vignolo et al. (2017), it tackles the task as a frame-by-frame binary classification using a kernel-based classifier. The second casts the problem as a sequence-to-sequence classification task

and employs a neural network model based on Gated Recurrent Units (GRU). We chose a GRU-based approach to leverage the natural structure of video frames: it is a parametric model that can exploit the inter-sample dependencies in sequence-structured data.

Kernel binary classification When considering the problem of biological motion detection as a frame-by-frame classification problem, the training set is the set couples of all the vectors defined in Equation (6.5) and their corresponding label, *i.e.* $\{x_i, y_i\}_{i=1}^{n}$ with n the total number of frames in the dataset. Here the main task consists in learning a function $f_k \colon \mathbb{R} \to \{-1, 1\}$ that is applied independently to each vector of a test sequence $\{x_t\}_{t=1}^{T}$ to predict whether the corresponding frames contain biological $(+1)$ or non-biological (-1) movement. For kernel-based models, it is possible to learn such functions by means of a closed form formula introduced in Section 1.6.4. For the reader convenience we recall briefly the form of a kernel based classifier and the learning formula:

$$f_k(x) = sgn\Big(\sum_{i=1}^{n} k(x, x_i)\, w_i\Big) \tag{6.6}$$

where $sgn \colon \mathbb{R} \to \{0, 1\}$ is the sign function, $k \colon \mathcal{X} \times \mathcal{X} \to \mathbb{R}$ is a positive definite function and $w \in \mathbb{R}^n$ is the set of weights that can be learnt from the data by solving

$$w = (K + \lambda n I_n)^{-1} \hat{Y} \tag{6.7}$$

with $K \in \mathbb{R}^{n \times n}$ the kernel matrix, $I_n \in \mathbb{R}^{n \times n}$ the identity matrix and $\hat{Y} = [y_1, \ldots, y_n]$ the vector containing all the labels. We make a remark, because Equation (6.6) is a non-parametric model, its size (the size of the vector w) and the cost for its evaluation grow as the cardinality of the dataset increases, specifically they both scale with $\mathcal{O}(n)$.

Gated Recurrent Unit Networks for Sequence Classification When recasting the problem as a sequence-to-sequence classification task the training set is the set sequences $\Big\{\big(\{s_t\}_{t=1}^{P}\big)_i, \big(\{y_t\}_{t=1}^{P}\big)_i\Big\}_{i=1}^{N}$ with N the total number of videos in the training set. A natural family of models to consider are recurrent neural networks, which we introduced in Section 2.3.2, specifically we consider Gated Recurrent Unit networks. These networks are based on a variant of RNN introduced in Chapter 2. We recall that recurrent models can be easily used to infer only one element at a a time with little computational cost. This allows for real-time applications where not all the elements of a sequence are at hand from the start, such as the task of biological motion detection.

Specifically for each element $s_t \in \mathbb{R}^d$ in a sequence, the GRU computes the following functions:

$$r_t = \sigma(W_r s_t + U_r m_{t-1}) \tag{6.8}$$

$$z_t = \sigma(W_z s_t + U_z m_{t-1}) \tag{6.9}$$

$$m_t = (1 - z_t) \odot m_{t-1} + z_t \odot \tanh(W_m s_t + U_m(r_t \odot m_{t-1})) \tag{6.10}$$

with $W_r, U_r, W_z, U_z, W_m, U_m \in \mathbb{R}^{d \times d}$, $\sigma \colon \mathbb{R}^d \to \mathbb{R}^d$ the element-wise sigmoid function and where $\odot$ is the Hadamard product. The matrices $W_r, U_r, W_z, U_z, W_m, U_m$ are learnt by means of gradient descent. For the sake of conciseness, we refer the interested reader to the works of Hochreiter and Schmidhuber (1997) and Chung et al. (2014) for additional details about this kind of networks. In our application we use a model composed of 5 Gated Recurrent Units sequentially stacked, each with hidden dimension $d = 32$. The features computed by the stacked GRUs are then fed to a linear layer followed by a hyperbolic tangent, mapping each element of the original sequence in the interval $[-1, 1]$. Values above 0 are classified as biological motions while values below are classified as non-biological. We choose this architecture because during our experiments we don't notice significant improvements with deeper or wider architectures and we observe a drop in test accuracy by using more shallow models.

6.3.3 Robotic implementation

As a byproduct of the experiment design process, we obtain an online application which we deploy on the R1 robot. Our robotics pipeline has the following structure:

1. we first extract the optical flow in the scene, by means of one of two different dense estimation methods, the classical algorithm introduced by Farnebäck (2003) or a state of art, deep learning-based approach (Sun et al., 2018)

2. using a simple tracking heuristic, tailored for this application, the robot understands when different entities are moving in the scene at the same time, so to compute their corresponding biological motion descriptions and to understand the evolution of their behavior over time

3. the trained model predicts the binary answer for the biological motion classification problem, separately for each entity which is moving

4. when there has been a biological entity moving in the scene for a while (we can specify this interval in seconds, for this step), the R1 robot is asked to fixate it, so to "focus its interest" on the corresponding part of the environment

As a sample use case of this application, we design a simple human-robot interaction scenario. The R1 robot is placed in front of a scene, facing a human subject and a non-biological actor. Within this scenario, the robot is asked to track the biological motions while ignoring non-biological sources of non-biological movement keeps moving, even in the case where both sources of movement are present in the scene. An example of this scenario of application can be found online[1]. The whole use case has been implemented by means of the YARP (Fitzpatrick et al., 2008) robotics framework.

6.4 Experiments

In this section we introduce the experimental results on our method compared to the baseline kernel classifier. To assess the quality of the GRU-based architecture we compare it with the kernel-based classifier. We train the kernel approach using both the features averaged over the last 5 frames and the features averaged with multiple filters up to a length of 30 frames as described in Vignolo et al. (2017). We choose a kernel classifier with Gaussian kernel $k(x,x') = exp\Big(-\frac{\|x-x'\|^2}{2\sigma^2}\Big)$ and select the best hyperparameters λ and σ using holdout cross-validation with a 50% random split average over 5 tries for each hyperparameter combination. We train our GRU-based architecture using Adam optimizer (Kingma and Ba, 2014) and select the best model using early stopping (Prechelt, 1998). We split the training set into a smaller training set and a validation set, we select the model achieving the best accuracy on the validation set. The validation set is comprised of one repetition of each combination *action-subject-view* for biological videos and *object-view* for non-biological videos. We then assess the prediction accuracy on the videos of the test set introduced in Section 6.2 and on a subset of videos previously removed from the training set: a recording for each base non-biological motion and a recording for each mixed video. It can be observed in Table 6.1 that the classifier using GRU outperforms the kernel counterpart on all tasks except for the old subject performing new actions, however in this case the performance is only slightly lower. For completeness we also show the performance of the original approach using a full 30 frameinterval of pre-processing.

[1] https://youtu.be/yLGnk5QXQBY

Table 6.1 Experimental results in terms of accuracy of the GRU-based approach and the kernel-based approach

	GRU (5)	Kernel(5)	Kernel (30)
Non-Bio Base Motions	**98.10**	86.68	95.37
Toy Train - Occluded	98.80	92.50	**100.0**
Fan rotating	98.97	98.12	**99.80**
Old Subject - New Actions	83.70	**84.47**	84.38
Old Subject - Occluded Base	**98.12**	87.28	97.69
New Subject - Base	91.38	83.27	**94.23**
New Subject - New	**99.23**	95.98	99.20
Mixed Videos	**98.40**	93.11	97.04
Total Average	95.83	90.17	**95.96**

Table 6.2 Experimental timing in milliseconds for inference of a single frame with kernel and GRU-based approach

	Per-frame time
Kernel model	7.99 ± 1.94
GRU model	0.79 ± 0.45

In Table 6.1 we show the average inference time per frame on a i7-8750H laptop CPU. Given the size of the dataset, the kernel-based classifier requires an amount of time one order of magnitude bigger than the GRU-based classifier, which has constant computational complexity for a fixed architecture.

Finally we investigate the quality of our approach and the kernel-based method by omitting completely the averaging step, *i.e.* using only the sequence of feature defined in Equation (6.5). In Table 6.3 both methods show, in general, accuracy values lower than

Table 6.3 Accuracy using the unfiltered features

	GRU	Kernel
Non-Bio Base Motions	**89.95**	78.80
Toy Train - Occluded	84.85	**97.33**
Fan rotating	**100.0**	96.58
Old Subject - New Actions	**93.51**	76.02
Old Subject - Occluded Base	**97.61**	77.98
New Subject - Base	**95.22**	70.39
New Subject - New	**99.45**	89.24
Mixed Videos	88.11	**89.30**
Total Average	**93.58**	84.45

those previously shown. And while the GRU model performs better than the kernel counterpart its results show more variance with respect to the same model trained using the averaged features.

Conclusions

We conclude this work summarizing the major contributions presented throughout the chapters and discussing future research direction. In this work we addressed the problem of structured prediction in machine learning, which consists in computing mappings from some input space to structured output domains such as differentiable manifolds, graphs, or sequences. After reviewing some of the main algorithms for structured prediction, we divide our efforts in two sections. In Part II we first establish a theoretical foundation for structured prediction on differentiable manifolds, and leverage it to develop a novel type of data representation. In Part III we proceed to specialize some of the structured algorithms for robotic oriented application, showing the improvements that can be obtained by accounting for the structure of the data.

More in detail, in Chapter 3 we study a structured prediction approach for manifold valued learning problems. This expands on the classical structured prediction algorithms that focus on finite size output domains. We characterize a wide class of loss functions (including the geodesic distance) for which we prove the considered algorithm to be statistically consistent, additionally providing finite sample bounds under standard regularity assumptions. Our experiments show promising results on synthetic and real data using two common manifolds: the positive definite matrices cone and the sphere. With the latter we considered applications on fingerprint reconstruction and multi-labeling. The proposed method leads to some open questions. From a statistical point of view it is of interest how invariants of the manifold explicitly affect the learning rates. From a more computational perspective, even if experimentally our algorithm achieves good results we did not investigate convergence guarantees in terms of optimization. Further developements would be directed at non-convex functional minimization on manifolds.

In Chapter 4 we show how to recast supervised problems with hierarchical structure as manifold-valued regressions in the hyperbolic manifold. We then propose two procedures for learning manifold-valued functions mapping from Euclidean to hyperbolic space: an extension of the algorithm proposed in Chapter 1, for which we also specialize general-

ization bounds, and a parametric deep-learning model that is informed by the geodesics of the output space. We evaluate both methods empirically on the task of hierarchical classification. Experiments show that hyperbolic structured prediction shows strong generalization performance. We also show that hyperbolic manifold regression enables new applications in supervised learning: by exploiting the continuous representation of hierarchies in hyperbolic space we were able to place unknown concepts in the embedding of a taxonomy using manifold regression. Moreover, by comparing to hyperbolic neural networks we showed that for this application, the key step is leveraging the geodesic of the manifold. In future work, we plan to expand this framework for more manifold-based data representations such as the sphere or product spaces. Furthermore we think that 0-shot learning applications could leverage this approach for hierarchical datasets.
In Chapter 5 we shift to a more applied problem for robotics. We consider the problem of learning the inverse kinematics with respect to both position and orientation for the robot end effector. We propose a consistent regularized structured predictor and validate this approach whitin the task of trajectory reconstruction. We compare it with an unstructured neural network as a baseline and with one class SVM, a method previously tested on this task. Our experiments show how consistent regularized structured predictors need less training points and smaller training time with respect to one class SVM. Moreover we show that for trajectories slightly outside of the robot workspace, the consistent regularized structured predictor manages to find an acceptable solution. This is in contrast to the unstructured neural network which shows a less precise behaviour. Possible developements include implementing our algorithms as a real-time application on a real mechanical arm or exploring possible path planning algorithms within the boundaries of the robot workspace.
In Chapter 6 we take a structured, deep learning based approach to the task of biological motion detection. This task consists in determining wether the movement present in a video stream is originated by a human or a non-biological entity. We use gated recurrent units networks, a type of structured neural networks, to recast a problem traditionally approached as a frame-by-frame classification task. By leveraging the sequence-like structure of data, we effectively remove 25 frames of latency introduced by the previous pipeline while still attaining almost state-of-the art accuracy. We also collect and distribute a dataset that can be used for other tasks of action recognition. Indeed, by leveraging more complex architectures or by using visual features computed by other deep models assigned to other task, it could be possible to refine the sequence classifier to accommodate for more complex task such as action classification or subject tracking. On the other hand,

possible developments would study alternative end-to-end architectures taking as input directly the video stream, in order to remove completely the intermediate step of optical flow extraction and temporal averaging.
This works shows new and promising ways of exploiting the structure in the output data for machine learning algorithms, and how this framework leads to many advantages. Ranging from a more principled approach in term of theory to higher performance in terms of computational complexity and accuracy.

References

Abadi, M., Barham, P., Chen, J., Chen, Z., Davis, A., Dean, J., Devin, M., Ghemawat, S., Irving, G., Isard, M., et al. (2016). Tensorflow: A system for large-scale machine learning. In *12th Symposium on Operating Systems Design and Implementation*, pages 265–283.

Absil, P.-A., Mahony, R., and Sepulchre, R. (2009). *Optimization algorithms on matrix manifolds*. Princeton University Press.

Alvarez, M. A., Rosasco, L., Lawrence, N. D., et al. (2012). Kernels for vector-valued functions: A review. *Foundations and Trends® in Machine Learning*, 4(3):195–266.

Amari, S.-i. and Nagaoka, H. (2007). *Methods of information geometry*, volume 191. American Mathematical Soc.

Andersen, M., Dahl, J., and Vandenberghe, L. (2013). Cvxopt: A python package for convex optimization. *abel. ee. ucla. edu/cvxopt*.

Aronszajn, N. (1950). Theory of reproducing kernels. *Transactions of the American mathematical society*, 68(3):337–404.

Bakir, G., Hofmann, T., Schölkopf, B., Smola, A., Taskar, B., and Vishwanathan, S. (2007). Predicting structured data. neural information processing.

Bauer, F., Pereverzev, S., and Rosasco, L. (2007). On regularization algorithms in learning theory. *Journal of complexity*, 23(1):52–72.

Bengio, Y., Courville, A., and Vincent, P. (2013). Representation learning: A review and new perspectives. *IEEE transactions on pattern analysis and machine intelligence*, 35(8):1798–1828.

Bhatia, R. (2009). *Positive definite matrices*. Princeton university press.

Bicer, V., Tran, T., and Gossen, A. (2011). Relational kernel machines for learning from graph-structured rdf data. In *Extended Semantic Web Conference*, pages 47–62. Springer.

Bisio, A., Sciutti, A., Nori, F., Metta, G., Fadiga, L., Sandini, G., and Pozzo, T. (2014). Motor contagion during human-human and human-robot interaction. *PloS one*, 9(8):e106172.

Blakemore, S.-J. and Decety, J. (2001). From the perception of action to the understanding of intention. *Nature reviews neuroscience*, 2(8):561.

Bócsi, B., Nguyen-Tuong, D., Csató, L., Schoelkopf, B., and Peters, J. (2011). Learning inverse kinematics with structured prediction. In *2011 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 698–703. IEEE.

Bojanowski, P., Grave, E., Joulin, A., and Mikolov, T. (2016). Enriching word vectors with subword information. *arXiv preprint arXiv:1607.04606*.

Bottou, L. and Bousquet, O. (2008). The tradeoffs of large scale learning. In *Advances in neural information processing systems*, pages 161–168.

Byrd, R. H., Lu, P., Nocedal, J., and Zhu, C. (1995). A limited memory algorithm for bound constrained optimization. *SIAM Journal on Scientific Computing*, 16(5):1190–1208.

Cai, L. and Hofmann, T. (2004). Hierarchical document categorization with support vector machines. In *Proceedings of the thirteenth ACM international conference on Information and knowledge management*, pages 78–87. ACM.

Calandriello, D., Carratino, L., Lazaric, A., Valko, M., and Rosasco, L. (2019). Gaussian process optimization with adaptive sketching: Scalable and no regret. *arXiv preprint arXiv:1903.05594*.

Calinon, S. (2018). Robot learning with task-parameterized generative models. In *Robotics Research*, pages 111–126. Springer.

Cauchy, A. (1847). Méthode générale pour la résolution des systemes d'équations simultanées. *Comp. Rend. Sci. Paris*, 25(1847):536–538.

Chamberlain, B. P., Hardwick, S. R., Wardrope, D. R., Dzogang, F., Daolio, F., and Vargas, S. (2019). Scalable hyperbolic recommender systems. *arXiv preprint arXiv:1902.08648*.

Chen, C. and Jackson, D. (2011). Parameterization and evaluation of robotic orientation workspace: A geometric treatment. *IEEE Transactions on Robotics*, 27(4):656–663.

Chen, L.-C., Zhu, Y., Papandreou, G., Schroff, F., and Adam, H. (2018). Encoder-decoder with atrous separable convolution for semantic image segmentation. In *Proceedings of the European conference on computer vision (ECCV)*, pages 801–818.

Chollet, F. (2016). Information-theoretical label embeddings for large-scale image classification. *arXiv preprint arXiv:1607.05691*.

Chua, K., Calandra, R., McAllister, R., and Levine, S. (2018). Deep reinforcement learning in a handful of trials using probabilistic dynamics models. In *Advances in Neural Information Processing Systems*, pages 4754–4765.

Chung, J., Gulcehre, C., Cho, K., and Bengio, Y. (2014). Empirical evaluation of gated recurrent neural networks on sequence modeling. *arXiv preprint arXiv:1412.3555*.

Ciliberto, C., Rosasco, L., and Rudi, A. (2016). A consistent regularization approach for structured prediction. *Advances in Neural Information Processing Systems 29 (NIPS)*, pages 4412–4420.

Ciliberto, C., Rudi, A., Rosasco, L., and Pontil, M. (2017). Consistent multitask learning with nonlinear output relations. In *Advances in Neural Information Processing Systems*, pages 1986–1996.

Colledanchise, M. and Ögren, P. (2018). *Behavior Trees in Robotics and AI: An Introduction*. CRC Press.

Correa, M., Hermosilla, G., Verschae, R., and Ruiz-del Solar, J. (2012). Human detection and identification by robots using thermal and visual information in domestic environments. *Journal of Intelligent & Robotic Systems*, 66(1-2):223–243.

Daume, H. C. and Marcu, D. (2006). *Practical structured learning techniques for natural language processing*. Citeseer.

Daumé III, H. (2004). From zero to reproducing kernel hilbert spaces in twelve pages or less. *University of Maryland*.

Davidson, T. R., Falorsi, L., De Cao, N., Kipf, T., and Tomczak, J. M. (2018). Hyperspherical variational auto-encoders. *arXiv preprint arXiv:1804.00891*.

De Angulo, V. R. and Torras, C. (2008). Learning inverse kinematics: Reduced sampling through decomposition into virtual robots. *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, 38(6):1571–1577.

De Sa, C., Gu, A., Ré, C., and Sala, F. (2018). Representation Tradeoffs for Hyperbolic Embeddings. In *International Conference on Machine Learning*, pages 4460–4469.

Deng, J., Dong, W., Socher, R., Li, L.-J., Li, K., and Fei-Fei, L. (2009). Imagenet: A large-scale hierarchical image database. In *2009 IEEE conference on computer vision and pattern recognition*, pages 248–255. Ieee.

Diestel, J. and Spalsbury, A. (2014). *The joys of Haar measure*. American Mathematical Soc.

D'Souza, A., Vijayakumar, S., and Schaal, S. (2001). Learning inverse kinematics. In *Proceedings 2001 IEEE/RSJ International Conference on Intelligent Robots and Systems. Expanding the Societal Role of Robotics in the the Next Millennium (Cat. No. 01CH37180)*, volume 1, pages 298–303. IEEE.

Duchi, J. C., Mackey, L. W., and Jordan, M. I. (2010). On the consistency of ranking algorithms. In *ICML*, pages 327–334.

Edunov, S., Ott, M., Auli, M., Grangier, D., and Ranzato, M. (2017). Classical structured prediction losses for sequence to sequence learning. *arXiv preprint arXiv:1711.04956*.

Falorsi, L., de Haan, P., Davidson, T. R., De Cao, N., Weiler, M., Forré, P., and Cohen, T. S. (2018). Explorations in homeomorphic variational auto-encoding. *arXiv preprint arXiv:1807.04689*.

Farasat, A., Nikolaev, A., Srihari, S. N., and Blair, R. H. (2015). Probabilistic graphical models in modern social network analysis. *Social Network Analysis and Mining*, 5(1):62.

Farnebäck, G. (2003). Two-frame motion estimation based on polynomial expansion. In *Scandinavian conference on Image analysis*, pages 363–370. Springer.

Farroni, T., Csibra, G., Simion, F., and Johnson, M. H. (2002). Eye contact detection in humans from birth. *Proceedings of the National Academy of Sciences*, 99(14):9602–9605.

Fitzpatrick, P., Metta, G., and Natale, L. (2008). Towards long-lived robot genes. *Robotics and Autonomous systems*, 56(1):29–45.

Fletcher, P. T. (2013). Geodesic regression and the theory of least squares on riemannian manifolds. *International journal of computer vision*, 105(2):171–185.

Fukushima, K. (1979). Neural network model for a mechanism of pattern recognition unaffected by shift in position-neocognitron. *IEICE Technical Report, A*, 62(10):658–665.

Gallot, S., Hulin, D., and Lafontaine, J. (1990). *Riemannian geometry*, volume 3. Springer.

Ganea, O.-E., Bécigneul, G., and Hofmann, T. (2018). Hyperbolic Neural Networks. In *Advances in neural information processing systems*, pages 5345–5355.

Gavazzi, G., Bisio, A., and Pozzo, T. (2013). Time perception of visual motion is tuned by the motor representation of human actions. *Scientific reports*, 3:1168.

Goodfellow, I., Bengio, Y., and Courville, A. (2016). *Deep learning*. MIT press.

Gower, J. C., Dijksterhuis, G. B., et al. (2004). *Procrustes problems*, volume 30. Oxford University Press on Demand.

Hahnloser, R. H., Sarpeshkar, R., Mahowald, M. A., Douglas, R. J., and Seung, H. S. (2000). Digital selection and analogue amplification coexist in a cortex-inspired silicon circuit. *Nature*, 405(6789):947.

Hamann, M. (2018). On the tree-likeness of hyperbolic spaces. *Mathematical Proceedings of the Cambridge Philosophical Society*, 164(2):345–361.

Härdle, W. and Simar, L. (2007). *Applied multivariate statistical analysis*, volume 22007. Springer.

Hauberg, S., Freifeld, O., and Black, M. J. (2012). A geometric take on metric learning. In *Advances in Neural Information Processing Systems*, pages 2024–2032.

He, K., Zhang, X., Ren, S., and Sun, J. (2016). Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778.

Hebey, E. (2000). *Nonlinear analysis on manifolds: Sobolev spaces and inequalities*, volume 5. American Mathematical Soc.

Hinkle, J., Muralidharan, P., Fletcher, P. T., and Joshi, S. (2012). Polynomial regression on riemannian manifolds. In *European Conference on Computer Vision*, pages 1–14. Springer.

Hochreiter, S. and Schmidhuber, J. (1997). Long short-term memory. *Neural computation*, 9(8):1735–1780.

Jae Hwang, S., Collins, M. D., Ravi, S. N., Ithapu, V. K., Adluru, N., Johnson, S. C., and Singh, V. (2015). A projection free method for generalized eigenvalue problem with a nonsmooth regularizer. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 1841–1849.

Jean, S., Cho, K., Memisevic, R., and Bengio, Y. (2014). On using very large target vocabulary for neural machine translation. *arXiv preprint arXiv:1412.2007*.

Joachims, T. (2002). Optimizing search engines using clickthrough data. In *Proceedings of the eighth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 133–142. ACM.

Joachims, T., Finley, T., and Yu, C.-N. J. (2009). Cutting-plane training of structural svms. *Machine Learning*, 77(1):27–59.

Jordan, M. I. and Rumelhart, D. E. (1992). Forward models: Supervised learning with a distal teacher. *Cognitive science*, 16(3):307–354.

Joulin, A., Grave, E., Bojanowski, P., and Mikolov, T. (2016). Bag of tricks for efficient text classification. *arXiv preprint arXiv:1607.01759*.

Kadous, M. W. and Sammut, C. (2005). Classification of multivariate time series and structured data using constructive induction. *Machine learning*, 58(2):179–216.

Khrulkov, V., Mirvakhabova, L., Ustinova, E., Oseledets, I., and Lempitsky, V. (2019). Hyperbolic image embeddings. *arXiv preprint arXiv:1904.02239*.

Kingma, D. P. and Ba, J. (2014). Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

Kingma, D. P. and Welling, M. (2013). Auto-encoding variational bayes. *arXiv preprint arXiv:1312.6114*.

Koppula, H. S., Gupta, R., and Saxena, A. (2013). Learning human activities and object affordances from rgb-d videos. *The International Journal of Robotics Research*, 32(8):951–970.

Krizhevsky, A., Sutskever, I., and Hinton, G. E. (2012). Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, pages 1097–1105.

Lacquaniti, F., Terzuolo, C., and Viviani, P. (1983). The law relating the kinematic and figural aspects of drawing movements. *Acta psychologica*, 54(1-3):115–130.

Lathuilière, S., Mesejo, P., Alameda-Pineda, X., and Horaud, R. (2019). A comprehensive analysis of deep regression. *IEEE transactions on pattern analysis and machine intelligence*.

Le, M., Roller, S., Papaxanthos, L., Kiela, D., and Nickel, M. (2019). Inferring concept hierarchies from text corpora via hyperbolic embeddings. *arXiv preprint arXiv:1902.00913*.

LeCun, Y., Bengio, Y., and Hinton, G. (2015). Deep learning. *nature*, 521(7553):436.

LeCun, Y. A., Bottou, L., Orr, G. B., and Müller, K.-R. (2012). Efficient backprop. In *Neural networks: Tricks of the trade*, pages 9–48. Springer.

Lee, J. M. (2003). Smooth manifolds. In *Introduction to Smooth Manifolds*, pages 1–29. Springer.

Loper, E. and Bird, S. (2002). NLTK. In *Proceedings of the ACL-02 Workshop on Effective tools and methodologies for teaching natural language processing and computational linguistics* -, volume 1, pages 63–70, Morristown, NJ, USA. Association for Computational Linguistics.

Lucchi, A., Li, Y., Smith, K., and Fua, P. (2012). Structured image segmentation using kernelized features. In *European Conference on Computer Vision*, pages 400–413. Springer.

Maiettini, E., Pasquale, G., Rosasco, L., and Natale, L. (2018). Speeding-up object detection training for robotics with falkon. In *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 5770–5776. IEEE.

Marconi, G. M., Ciliberto, C., and Rosasco, L. (2020). Hyperbolic manifold regression. In *Proceedings of the 23th International Conference on Artificial Intelligence and Statistics*.

Martin, G. J. (1989). Balls in hyperbolic manifolds. *Journal of the London mathematical society*, 2(2):257–264.

McCulloch, W. S. and Pitts, W. (1943). A logical calculus of the ideas immanent in nervous activity. *The bulletin of mathematical biophysics*, 5(4):115–133.

Metta, G., Natale, L., Nori, F., Sandini, G., Vernon, D., Fadiga, L., von Hofsten, C., Rosander, K., Lopes, M., Santos-Victor, J., Bernardino, A., and Montesano, L. (2010). The icub humanoid robot: an open-systems platform for research in cognitive development. *Neural networks : the official journal of the International Neural Network Society*, 23(8-9):1125–34.

Micchelli, C. A. and Pontil, M. (2005). On learning vector-valued functions. *Neural computation*, 17(1):177–204.

Mikolov, T., Sutskever, I., Chen, K., Corrado, G., and Dean, J. (2013). Distributed representations of words and phrases and their compositionality. *CoRR*, abs/1310.4546.

Minh, H. Q. and Murino, V. (2017). Covariances in computer vision and machine learning. *Synthesis Lectures on Computer Vision*, 7(4):1–170.

Moakher, M. and Batchelor, P. G. (2006). Symmetric positive-definite matrices: From geometry to applications and visualization. In *Visualization and Processing of Tensor Fields*, pages 285–298. Springer.

Morris, J. S. (2015). Functional regression. *Annual Review of Statistics and Its Application*, 2:321–359.

Mrčun, J. (2005). On isomorphisms of algebras of smooth functions. *Proceedings of the American Mathematical Society*, 133(10):3109–3113.

Mutlu, B., Kanda, T., Forlizzi, J., Hodgins, J., and Ishiguro, H. (2012). Conversational gaze mechanisms for humanlike robots. *ACM Transactions on Interactive Intelligent Systems (TiiS)*, 1(2):12.

Nestruev, J. (2006). *Smooth manifolds and observables*, volume 220. Springer Science & Business Media.

Ngiam, J., Khosla, A., Kim, M., Nam, J., Lee, H., and Ng, A. Y. (2011). Multimodal deep learning. In *Proceedings of the 28th international conference on machine learning (ICML-11)*, pages 689–696.

Nickel, M. and Kiela, D. (2017). Poincaré Embeddings for Learning Hierarchical Representations. In *Neural Information Processing Systems Proceedings*, pages 6338–6347.

Nickel, M. and Kiela, D. (2018). Learning Continuous Hierarchies in the Lorentz Model of Hyperbolic Geometry. In *International Conference on Machine Learning*.

Nickel, M., Tresp, V., and Kriegel, H. (2011). A three-way model for collective learning on multi-relational data. In *Proceedings of the 28th International Conference on Machine Learning*, pages 809–816.

Nielsen, F. and Sun, K. (2017). Clustering in hilbert simplex geometry. *arXiv preprint arXiv:1704.00454*.

Nocedal, J. and Wright, S. (2006). Numerical optimization: Springer science & business media. *New York*.

Noceti, N., Sciutti, A., and Sandini, G. (2015). Cognition helps vision: Recognizing biological motion using invariant dynamic cues. In *International Conference on Image Analysis and Processing*, pages 676–686. Springer.

Nowozin, S., Lampert, C. H., et al. (2011). Structured learning and prediction in computer vision. *Foundations and Trends® in Computer Graphics and Vision*, 6(3–4):185–365.

Okuno, A., Kim, G., and Shimodaira, H. (2018). Graph embedding with shifted inner product similarity and its improved approximation capability. *arXiv preprint arXiv:1810.03463*.

Oyama, E., Chong, N. Y., Agah, A., and Maeda, T. (2001). Inverse kinematics learning by modular architecture neural networks with performance prediction networks. In *Proceedings 2001 ICRA. IEEE International Conference on Robotics and Automation (Cat. No. 01CH37164)*, volume 1, pages 1006–1012. IEEE.

Oyama, E., Maeda, T., Gan, J. Q., Rosales, E. M., MacDorman, K. F., Tachi, S., and Agah, A. (2005). Inverse kinematics learning for robotic arms with fewer degrees of freedom by modular neural network systems. In *2005 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 1791–1798. IEEE.

Paaßen, B., Göpfert, C., and Hammer, B. (2017). Time series prediction for graphs in kernel and dissimilarity spaces. *Neural Processing Letters*, pages 1–21.

Park, F. C. and Kim, J. W. (1998). Manipulability and singularity analysis of multiple robot systems: A geometric approach. In *Proceedings. 1998 IEEE International Conference on Robotics and Automation (Cat. No. 98CH36146)*, volume 2, pages 1032–1037. IEEE.

Parmiggiani, A., Fiorio, L., Scalzo, A., Sureshbabu, A. V., Randazzo, M., Maggiali, M., Pattacini, U., Lehmann, H., Tikhanoff, V., Domenichelli, D., Cardellino, A., Congiu, P., Pagnin, A., Cingolani, R., Natale, L., and Metta, G. (2017). The design and validation of the r1 personal humanoid. In *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 674–680.

Paszke, A., Gross, S., Chintala, S., Chanan, G., Yang, E., DeVito, Z., Lin, Z., Desmaison, A., Antiga, L., and Lerer, A. (2017). Automatic differentiation in pytorch.

Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., Vanderplas, J., Passos, A., Cournapeau, D., Brucher, M., Perrot, M., and Duchesnay, É. (2011). Scikit-learn: Machine Learning in Python. *Journal of Machine Learning Research*, 12(Oct):2825–2830.

Pelossof, R., Miller, A., Allen, P., and Jebara, T. (2004). An svm learning approach to robotic grasping. In *IEEE International Conference on Robotics and Automation, 2004. Proceedings. ICRA'04. 2004*, volume 4, pages 3512–3518. IEEE.

Perozzi, B., Al-Rfou, R., and Skiena, S. (2014). Deepwalk: Online learning of social representations. In *Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 701–710. ACM.

Peypouquet, J. (2015). *Convex optimization in normed spaces: theory, methods and examples*. Springer.

Peyré, G., Cuturi, M., et al. (2019). Computational optimal transport. *Foundations and Trends® in Machine Learning*, 11(5-6):355–607.

Pozzo, T., Papaxanthis, C., Petit, J. L., Schweighofer, N., and Stucchi, N. (2006). Kinematic features of movement tunes perception and action coupling. *Behavioural brain research*, 169(1):75–82.

Prechelt, L. (1998). Early stopping-but when? In *Neural Networks: Tricks of the trade*, pages 55–69. Springer.

Pucci, D., Nava, G., and Nori, F. (2016). Automatic gain tuning of a momentum based balancing controller for humanoid robots. In *2016 IEEE-RAS 16th International Conference on Humanoid Robots (Humanoids)*, pages 158–164. IEEE.

Radford, A., Metz, L., and Chintala, S. (2015). Unsupervised representation learning with deep convolutional generative adversarial networks. *arXiv preprint arXiv:1511.06434*.

Robbin, J. W. and Salamon, D. A. (2011). Introduction to differential geometry. *ETH, Lecture Notes, preliminary version, January*.

Rudi, A., Ciliberto, C., Marconi, G. M., and Rosasco, L. (2018). Manifold structured prediction. In *Advances in Neural Information Processing Systems*, pages 5610–5621.

Rumelhart, D. E., Hinton, G. E., Williams, R. J., et al. (1988). Learning representations by back-propagating errors. *Cognitive modeling*, 5(3):1.

Sakai, T. (1996). *Riemannian geometry*, volume 149. American Mathematical Soc.

Sattar, J. and Dudek, G. (2018). Visual identification of biological motion for underwater human–robot interaction. *Autonomous Robots*, 42(1):111–124.

Schmidhuber, J. (1992). Learning complex, extended sequences using the principle of history compression. *Neural Computation*, 4(2):234–242.

Schölkopf, B., Platt, J. C., Shawe-Taylor, J., Smola, A. J., and Williamson, R. C. (2001). Estimating the support of a high-dimensional distribution. *Neural computation*, 13(7):1443–1471.

Scholkopf, B. and Smola, A. J. (2001). *Learning with kernels: support vector machines, regularization, optimization, and beyond*. MIT press.

Sciutti, A., Bisio, A., Nori, F., Metta, G., Fadiga, L., Pozzo, T., and Sandini, G. (2012). Measuring human-robot interaction through motor resonance. *International Journal of Social Robotics*, 4(3):223–234.

Selig, J. M. (2013). *Geometrical methods in robotics*. Springer Science & Business Media.

Shor, N. Z. (2012). *Minimization methods for non-differentiable functions*, volume 3. Springer Science & Business Media.

Siciliano, B. (1990). Kinematic control of redundant robot manipulators: A tutorial. *Journal of intelligent and robotic systems*, 3(3):201–212.

Simion, F., Regolin, L., and Bulf, H. (2008). A predisposition for biological motion in the newborn baby. *Proceedings of the National Academy of Sciences*, 105(2):809–813.

Song, L., Fukumizu, K., and Gretton, A. (2013). Kernel embeddings of conditional distributions: A unified kernel framework for nonparametric inference in graphical models. *IEEE Signal Processing Magazine*, 30(4):98–111.

Spong, M. W. (1992). Remarks on robot dynamics: canonical transformations and riemannian geometry. In *Proceedings 1992 IEEE International Conference on Robotics and Automation*, pages 554–559. IEEE.

Spong, M. W., Hutchinson, S., Vidyasagar, M., et al. (2006). *Robot modeling and control*, volume 3. wiley New York.

Srinivasan, A. (1999). Note on the location of optimal classifiers in n-dimensional roc space.

Steinke, F. and Hein, M. (2009). Non-parametric regression between manifolds. In *Advances in Neural Information Processing Systems*, pages 1561–1568.

Steinke, F., Hein, M., and Schölkopf, B. (2010). Nonparametric regression between general riemannian manifolds. *SIAM Journal on Imaging Sciences*, 3(3):527–563.

Steinwart, I. and Christmann, A. (2008). *Support vector machines*. Springer Science & Business Media.

Strapparava, C., Valitutti, A., et al. (2004). Wordnet affect: an affective extension of wordnet. In *Lrec*, volume 4, page 40. Citeseer.

Sun, D., Yang, X., Liu, M.-Y., and Kautz, J. (2018). Pwc-net: Cnns for optical flow using pyramid, warping, and cost volume. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 8934–8943.

Sung, J., Ponce, C., Selman, B., and Saxena, A. (2012). Unstructured human activity detection from rgbd images. In *2012 IEEE international conference on robotics and automation*, pages 842–849. IEEE.

Takeuchi, I., Le, Q. V., Sears, T. D., and Smola, A. J. (2006). Nonparametric quantile estimation. *Journal of machine learning research*, 7(Jul):1231–1264.

Tay, Y., Tuan, L. A., and Hui, S. C. (2017). Hyperbolic Representation Learning for Fast and Efficient Neural Question Answering.

Tevatia, G. and Schaal, S. (2000). Inverse kinematics for humanoid robots. In *Proceedings 2000 ICRA. Millennium Conference. IEEE International Conference on Robotics and Automation. Symposia Proceedings (Cat. No. 00CH37065)*, volume 1, pages 294–299. IEEE.

Thrun, S. et al. (2002). Robotic mapping: A survey. *Exploring artificial intelligence in the new millennium*, 1(1-35):1.

Tifrea, A., Bécigneul, G., and Ganea, O.-E. (2018). Poincar\'e glove: Hyperbolic word embeddings. *arXiv preprint arXiv:1810.06546*.

Tosun, T., Mead, R., and Stengel, R. (2014). A general method for kinematic retargeting: Adapting poses between humans and robots. In *Proc. ASME Int. Mech. Eng. Congress and Exposition*, pages 1–10. Citeseer.

Treves, F. (2016). *Topological Vector Spaces, Distributions and Kernels: Pure and Applied Mathematics*, volume 25. Elsevier.

Tsay, R. S. (2014). Financial time series. *Wiley StatsRef: Statistics Reference Online*, pages 1–23.

Tsoumakas, G., Katakis, I., and Vlahavas, I. (2009). Mining multi-label data. In *Data mining and knowledge discovery handbook*, pages 667–685. Springer.

Ungar, A. A. (2008). *Analytic hyperbolic geometry and Albert Einstein's special theory of relativity*. World scientific.

Vandenberghe, L. (2010). The cvxopt linear and quadratic cone program solvers. *Online: http://cvxopt. org/documentation/coneprog. pdf*.

Veit, A., Nickel, M., Belongie, S., and van der Maaten, L. (2018). Separating self-expression and visual content in hashtag supervision. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 5919–5927.

Vignolo, A., Noceti, N., Rea, F., Sciutti, A., Odone, F., and Sandini, G. (2017). Detecting biological motion for human–robot interaction: A link between perception and action. *Frontiers in Robotics and AI*, 4:14.

Villani, C. (2008). *Optimal transport: old and new*, volume 338. Springer Science & Business Media.

Viviani, P., Baud-Bovy, G., and Redolfi, M. (1997). Perceiving and tracking kinesthetic stimuli: Further evidence of motor–perceptual interactions. *Journal of Experimental Psychology: Human Perception and Performance*, 23(4):1232.

Weng, J. J., Ahuja, N., and Huang, T. S. (1993). Learning recognition and segmentation of 3-d objects from 2-d images. In *1993 (4th) International Conference on Computer Vision*, pages 121–128. IEEE.

Werbos, P. (1974). Beyond regression:" new tools for prediction and analysis in the behavioral sciences. *Ph. D. dissertation, Harvard University*.

Westervelt, E. R., Grizzle, J. W., Chevallereau, C., Choi, J. H., and Morris, B. (2018). *Feedback control of dynamic bipedal robot locomotion*. CRC press.

Whitley, D. and Watson, J. (1970). *Complexity Theory and the No Free Lunch Theorem*, pages 317–339.

Wilson, R. C., Hancock, E. R., Pekalska, E., and Duin, R. P. (2014). Spherical and hyperbolic embeddings of data. *IEEE transactions on pattern analysis and machine intelligence*, 36(11):2255–2269.

Wolter, F.-E. (1979). Distance function and cut loci on a complete riemannian manifold. *Archiv der Mathematik*, 32(1):92–96.

Xiao, T., Li, H., Ouyang, W., and Wang, X. (2016). Learning deep feature representations with domain guided dropout for person re-identification. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1249–1258.

Xu, J. and Durrett, G. (2018). Spherical latent spaces for stable variational autoencoders. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 4503–4513.

Yao, K. (1967). Applications of reproducing kernel hilbert spaces–bandlimited signal models. *Information and Control*, 11(4):429–444.

Zhang, H. and Sra, S. (2016). First-order methods for geodesically convex optimization. In *Conference on Learning Theory*, pages 1617–1638.

Zhang, T. et al. (2004). Statistical behavior and consistency of classification methods based on convex risk minimization. *The Annals of Statistics*, 32(1):56–85.

Appendix A

Appendix

A.1 Proof of Theorem 3.4.1

We prove here intermediate results that will be key to prove Theorem 3.4.1. We refer to Lee (2003) for basic definitions on manifolds and to Aronszajn (1950) for an introduction on reproducing kernel Hilbert spaces (RKHS).

Notation and Definitions. We recall here basic notations and definition that will be used in the following. Given a smooth manifold $\mathcal{M}$, for any open subset $U \subseteq \mathcal{M}$ we denote by $C^\infty(U)$ the set of smooth functions on U and with $C_c^\infty(U)$ the set of *compactly supported* smooth functions on U, namely functions such that the closure of their support is a compact set. For a compact subset $N \subset \mathcal{M}$ we denote by $C_c^\infty(N)$ the set of all functions $h : N \to \mathbb{R}$ that admit an extension $\bar{h} \in C_c^\infty(\mathcal{M})$ such that $\bar{h}|_N = h$ and its support is contained in N, namely it vanishes on the border of N. Finally, for any subset N of $\mathcal{M}$ we denote $C^\infty(N)$ the set of all functions that admit a smooth extension in $C^\infty(\mathcal{M})$.
In the following, a central role will be played by tensor product of topological vector spaces Treves (2016). In particular, for a Hilbert space $\mathcal{H}$, we will denote $\mathcal{H} \otimes \mathcal{H}$ the closure of the tensor product between $\mathcal{H}$ and itself with respect to the canonical norm such that $\|h \otimes h'\|_{\mathcal{H}\otimes\mathcal{H}} = \|h\|_{\mathcal{H}}\|h'\|_{\mathcal{H}}$ for any $h, h' \in \mathcal{H}$. Moreover, to given a compact set $N \subset \mathbb{R}^d$, we recall that $C_c^\infty(N)\hat{\otimes}_\pi C_c^\infty(N)$ denotes the completion of the topological tensor product between $C_c^\infty(N)$ and itself with respect to the projective topology (see Treves (2016) Def. 43.2 and 43.5). In the following, for simplicity, we will denote this space with $C_c^\infty(N) \otimes C_c^\infty(N)$ with some abuse of notation. Finally, for any subset $\mathcal{Y} \subseteq \mathcal{M}$ and space $\mathcal{F}$ of functions from $\mathcal{M}$ to $\mathbb{R}$ we denote by $\mathcal{F}|_{\mathcal{Y}}$ the space of functions from $\mathcal{Y}$ to $\mathbb{R}$ that admit an extension in $\mathcal{F}$. In particular not that $C^\infty(\mathcal{Y}) = C^\infty(\mathcal{M})|_{\mathcal{Y}}$.

A.1.1 Auxiliary Results

We are ready to prove the auxiliary results.

Lemma A.1.1. *Let $\mathcal{M}$ be a topological space, $Y \subseteq \mathcal{M}$ be a compact subset and $\mathcal{H}$ a reproducing kernel Hilbert space of functions on $\mathcal{M}$ with kernel $K : \mathcal{M} \times \mathcal{M} \to \mathbb{R}$ such that there exists $\kappa > 0$ for which $k(y,y) \leq \kappa^2$ for any $y \in \mathcal{Y}$. Then, for any $\bar{h} \in \mathcal{H} \otimes \mathcal{H}$, its restriction to $\mathcal{Y} \times \mathcal{Y}$, $h = \bar{h}|_{\mathcal{Y}\times\mathcal{Y}}$ is SELF.*

Proof. Denote $K_y = k(y,\cdot) \in \mathcal{H}$ for every $y \in \mathcal{M}$. Then the space $\mathcal{H} \otimes \mathcal{H}$ is an RKHS with reproducing kernel $\bar{K} : (\mathcal{M}\times\mathcal{M})\times(\mathcal{M}\times\mathcal{M}) \to \mathbb{R}$ such that $\bar{K}((y,z),(y',z')) = K(y,y')K(z,z')$ for any $y, y', z, z' \in \mathcal{M}$ (see e.g. Aronszajn (1950)). In particular $\bar{K}_{(y,z)} = K_y \otimes K_z$. Let now $\bar{h} : \mathcal{M} \times \mathcal{M} \to \mathbb{R}$ be a function in $\mathcal{H} \otimes \mathcal{H}$. In particular, there exist a $V \in \mathcal{H} \otimes \mathcal{H}$ such that $\langle V, K_y \otimes K_z \rangle_{\mathcal{H}\otimes\mathcal{H}} = \bar{h}(y,z)$ for any $y, z \in \mathcal{Y}$ (reproducing property). Note that $\mathcal{H} \otimes \mathcal{H}$ is isometric to the space of Hilbert-Schmidt operators from $\mathcal{H}$ to itself, with inner product corresponding to $\langle A, B\rangle_{\mathcal{H}\otimes\mathcal{H}} = \langle A, B\rangle_{\mathrm{HS}} = \mathrm{Tr}(A^* B)$ for any $A, B \in \mathcal{H} \otimes \mathcal{H}$, with A^* denoting the conjugate of $A^* \in \mathcal{H} \otimes \mathcal{H}$. Therefore, for any $y, z \in \mathcal{Y}$ we have

$$\begin{aligned} \bar{h}|_{\mathcal{Y}\times\mathcal{Y}}(y,z) &= \bar{h}(y,z) \\ &= \langle V, K_y \otimes K_z \rangle_{\mathcal{H}\otimes\mathcal{H}} \\ &= \mathrm{Tr}(V^* K_y \otimes K_z) \end{aligned}$$

and thus

$$h|_{\mathcal{Y}\times\mathcal{Y}}(y,z) = \langle K_z, V^* K_y \rangle_{\mathcal{H}}. \tag{A.1}$$

Since K_y is bounded in $\mathcal{H}$, for $y \in \mathcal{Y}$ and the operator norm of V is bounded by its Hilbert-Schmidt norm, namely $\|V\| \leq \|V\|_{\mathrm{HS}}$, we can conclude that $h = \bar{h}|_{\mathcal{Y}\times\mathcal{Y}}$ is indeed SELF. □

Lemma A.1.2. *Let $\mathcal{M}$ satisfy Assumption 3.4.1. Then there exists a reproducing kernel Hilbert space of functions $\mathcal{H}$ on $\mathcal{M}$, with bounded kernel, such that $C_c^\infty(\mathcal{M}) \subseteq \mathcal{H}$.*

Proof. Let $H_s^2(\mathcal{M})$ denote the Sobolev space on $\mathcal{M}$ of squared integrable functions with smoothness $s > 0$ (see Hebey (2000) for the definition of Sobolev spaces on Riemannian manifolds). By construction (see page 47 of Hebey (2000)), $C_c^\infty(\mathcal{M}) \subset H_s^2(\mathcal{M})$ for any $s > 0$. To prove this Lemma, we will show that $H_s^2(\mathcal{M})$ is an RKHS for any $s > d/2$. The proof is organized in two steps.

Step 1: $H_s^2(\mathcal{M})$ is continuously embedded in $C(\mathcal{M})$. By Assumption 3.4.1, we can apply Thm. 3.4 in Hebey (2000) (see also Thm. 2.7 Hebey (2000) for compact manifolds), which

guarantees the existence of a constant $C > 0$ (see last lines of the proofs for its explicit definition) such that

$$\sup_{y \in \mathcal{M}} |f(y)| \leq C \|f\|_{\mathcal{H}_s^2(\mathcal{M})},$$

for any $y \in \mathcal{M}$ and $f \in \mathcal{H}_s^2(\mathcal{M})$.

Step 2: Constructing $\mathcal{H}$ from $H_s^2(\mathcal{M})$. Prop. 2.1 of Hebey (2000) proves that there exists an inner product, that we denote by $\langle \cdot, \cdot \rangle_{\mathcal{H}}$, whose associated norm is equivalent to $\| \cdot \|_{H_s^2(\mathcal{M})}$ and such that the space $\mathcal{H} = (H_s^2(\mathcal{M}), \langle \cdot, \cdot \rangle_{\mathcal{H}})$ is a Hilbert space.
Now, for any $y \in \mathcal{M}$ denote by $e_y : \mathcal{H} \to \mathbb{R}$, the linear functional corresponding to the evaluation, that is $e_y(f) = f(y)$. Now by Step 1, we have that the linear functional e_y is uniformly bounded and so continuous, indeed,

$$|e_y(f)| = |f(y)| \leq C \|f\|_{\mathcal{H}}, \quad \forall f \in \mathcal{H}.$$

So by the Riesz representation theorem $e_y \in \mathcal{H}$ and so $\mathcal{H}$ is a reproducing kernel Hilbert space, with kernel $k(y, y') = \langle e_y, e_{y'} \rangle_{\mathcal{H}}$, (see Aronszajn (1950), page 343, for more details). Note finally that the kernel is bounded since

$$\|e_y\|_{\mathcal{H}} = \sup_{\|f\|_{\mathcal{H}} \leq 1} |\langle e_y, f \rangle_{\mathcal{H}}| = \sup_{\|f\|_{\mathcal{H}} \leq 1} |e_y(f)| \leq C,$$

and therefore $k(y, y') \leq \|e_y\|_{\mathcal{H}} \|e_{y'}\|_{\mathcal{H}} \leq C^2$. □

In the following, let $A \subseteq \{f : U \to S\}$ and $B \subseteq \{g : V \to S\}$, with U, V, S topological spaces. We denote $A \cong B$ if there exists an invertible map $q : U \to V$, such that $B = A \circ q^{-1}$ and $A = B \circ q$.

Lemma A.1.3 (see also Mrčun (2005); Nestruev (2006))**.** *Let U be a geodesically convex open subset of a d-dimensional complete Riemannian manifold $\mathcal{M}$ without border, then there exists a smooth map $q : U \to \mathbb{R}^d$ with smooth inverse, such that*

$$C^\infty(U) \cong C^\infty(\mathbb{R}^d), \qquad \textit{and} \qquad C_c^\infty(U) \cong C_c^\infty(\mathbb{R}^d)$$

moreover for any compact set $\mathcal{Y} \subset U$ there exists a compact set $R \subset \mathbb{R}^d$ such that $R = q(\mathcal{Y})$ and the map s, that is the restriction of q to $\mathcal{Y} \to R$, guarantees

$$C^\infty(\mathcal{Y}) \cong C^\infty(R), \qquad \textit{and} \qquad C_c^\infty(\mathcal{Y}) \cong C_c^\infty(R)$$

Proof. By Lemma A.2.1, there exists a point $p \in U$ such that $d(p,\cdot)$ admits all directional derivatives in all points $q \in U$ (it is, in fact in $C^\infty(U)$). We are therefore in the hypotheses of Thm. 2 in Wolter (1979), from which we conclude that there exists a smooth diffeomorphism between U and $\mathbb{R}^d$ (with smooth inverse). Denoting by q the diffeomorphism between U and $\mathbb{R}^d$, for any function $f \in C^\infty(U)$, we have $f \circ q^{-1} \in C^\infty(\mathbb{R}^d)$, so $C^\infty(U) \circ q^{-1} \subseteq C^\infty(\mathbb{R}^d)$ and for any function $g \in C^\infty(\mathbb{R}^d)$ we have $g \circ q \in C^\infty(U)$, so $C^\infty(\mathbb{R}^d) \circ q \subseteq C^\infty(U)$. Finally we recall that if $A \subseteq B$, then $A \circ p \subseteq B \circ p$ for any set A, B and any map p applicable to A, B. Then

$$C^\infty(U) = C^\infty(U) \circ q^{-1} \circ q \subseteq C^\infty(\mathbb{R}^d) \circ q \subseteq C^\infty(U)$$

and so $C^\infty(N) \cong C^\infty(\mathbb{R}^d)$. The same reasoning holds $C_c^\infty(U) \cong C_c^\infty(\mathbb{R}^d)$.
Analogously, the smooth diffeomorphism q maps compact subsets of U to compact subsets of $\mathbb{R}^d$. Denote by $R \subset \mathbb{R}^d$ the compact subset that is $q(\mathcal{Y})$, the image of $\mathcal{Y} \subseteq U$ a compact subset of U, then s is the restriction of q to $\mathcal{Y} \to R$. By the same reasoning as above, we have that $C^\infty(\mathcal{Y}) \cong C^\infty(R)$ via s. □

Lemma A.1.4. *Let U be a open geodesically convex subset of a complete Riemannian d-dimensional manifold $\mathcal{M}$ and $\mathcal{Y}$ a compact subset of U, then there exists a compact subset $N \subseteq U$ such that $\mathcal{Y}$ belongs to the interior of N and*

$$C^\infty(\mathcal{Y} \times \mathcal{Y}) \subseteq (C_c^\infty(N) \otimes C_c^\infty(N))|_{Y \times Y}.$$

Moreover, $C^\infty(\mathcal{Y}) \subseteq C_c^\infty(N)|_{\mathcal{Y}}$.

Proof. We first consider the real case $U = \mathcal{M} = \mathbb{R}^d$ with Euclidean metric. By Cor. 2.19 in Lee (2003), for any open subset $V \subset \mathbb{R}^d$ we have that any $f \in C^\infty(\mathcal{Y})$ admits an extension $\tilde{f} \in C^\infty(\mathbb{R}^d)$ such that $\tilde{f}|_{\mathcal{Y}} = f$ and $\operatorname{supp}\tilde{f} \subset C_c^\infty(V)$. Then, since $\mathcal{Y}$ is bounded (compact in a complete space), there exists a bounded open set V containing $\mathcal{Y}$. Let $N = \overline{V}$ the closure of V. N is a compact set as well and contains $\mathcal{Y}$ in its interior. In particular, since for any $f \in C^\infty(\mathcal{Y})$ the extension $\tilde{f}$ has support contained in $V \subset N$, this shows that $C^\infty(\mathcal{Y}) \subseteq C_c^\infty(N)$. Analogously we have $C^\infty(\mathcal{Y} \times \mathcal{Y}) \subseteq C_c^\infty(N \times N)$.
Now, by Thm. 51.6 (a) in Treves (2016), we have that

$$C_c^\infty(N) \otimes C_c^\infty(N) \cong C_c^\infty(N \times N).$$

which concludes the proof in the real setting. The proof generalizes trivially to the case where U is an open geodesically convex subset of a complete Riemannian manifold thanks to the isomorphisms between spaces of smooth functions provided by Lemma A.1.3. □

A.1.2 Proof of Theorem 3.4.1

For the following results we need to introduce the concept of *cut locus*. For any $y \in \mathcal{M}$, denote by $\mathrm{Cut}(y) \subseteq \mathcal{M}$ the *cut locus* of y the closure of the set of points $z \in \mathcal{M}$ that are connected to y by more than one minimal geodesic (see Gallot et al. (1990); Sakai (1996)). For any $y \in \mathcal{Y}$ we have $y \in \mathcal{M} \setminus \mathrm{Cut}(y)$, see e.g. Lemma 4.4 in Sakai (1996).
Finally we refine Assumption 3.4.2 to avoid pathological cases. Indeed a geodesically convex set can still have conjugate points on the boundary. To avoid this situation we restate Assumption 3.4.2 as follows

Assumption 2' *$\widetilde{M}$ is an open geodesically convex subset of the manifold $\mathcal{M}$ and $\mathcal{Y}$ is a compact subset of $\widetilde{M}$.*

Proof of Theorem 3.4.1. By Asm. 2', let $\widetilde{\mathcal{M}}$ be an open geodesically convex subset of $\mathcal{M}$ such that $\mathcal{Y} \subset \widetilde{\mathcal{M}} \subseteq \mathcal{M}$. Apply Lemma A.1.4 and let $N \subseteq \widetilde{\mathcal{M}}$ be a compact set such that $\mathcal{Y}$ is contained in the interior of N, namely

$$C^\infty(\mathcal{Y}) \subseteq C_c^\infty(N)|_{\mathcal{Y}} \subseteq C_c^\infty(\mathcal{M})|_{\mathcal{Y}} \subseteq \mathcal{H}|_{\mathcal{Y}}.$$

Then, by applying again Lemma A.1.4 we have

$$C^\infty(\mathcal{Y} \times \mathcal{Y}) \subseteq (C_c^\infty(N) \otimes C_c^\infty(N))|_{\mathcal{Y}\times\mathcal{Y}} \subseteq (\mathcal{H} \otimes \mathcal{H})|_{\mathcal{Y}\times\mathcal{Y}}. \tag{A.2}$$

Therefore we conclude that for any $h \in C^\infty(\mathcal{Y})$, there exists $\bar{h} : \mathcal{M} \times \mathcal{M} \to \mathbb{R}$ with $\bar{h} \in \mathcal{H} \otimes \mathcal{H}$ and $h = \bar{h}|_{\mathcal{Y}\times\mathcal{Y}}$. Finally we apply Lemma A.1.1 to $\bar{h}$, which guarantees h to be SELF. □

A.2 Proof of Theorem 3.4.2

We prove a preliminary result.

Lemma A.2.1. *Let $\mathcal{M}$ be a Riemannian manifold and N be a geodesically convex subset of $\mathcal{M}$. Then,*

$$d^2|_{N\times N} \in C^\infty(N \times N).$$

Proof. For any $y \in \mathcal{M}$, denote $\mathrm{Cut}(y) \subseteq \mathcal{M}$ the *cut locus* of y, that is the set of points in $z \in \mathcal{M}$ that are connected by more than one minimal geodesic curve with y (see Gallot et al. (1990); Sakai (1996)). Let $\mathrm{Cut}(\mathcal{M}) = \bigcup_{y\in\mathcal{M}}(\{y\} \times \mathrm{Cut}(y)) \subseteq \mathcal{M} \times \mathcal{M}$. Then, then the squared geodesic distance is such that (see e.g. Villani (2008), page 336)

$$d^2 \in C^\infty(\mathcal{M} \times \mathcal{M} \setminus \mathrm{Cut}(\mathcal{M})).$$

Now note that by definition of geodesically convex subset $N \subseteq \mathcal{M}$, for any two points in N there exist one and only one minimizing geodesic curve connecting them. Therefore, $N \times N \cap \mathrm{Cut}(\mathcal{M}) = \emptyset$ and consequently $N \times N \subseteq \mathcal{M} \times \mathcal{M} \setminus \mathrm{Cut}(\mathcal{M})$. We conclude that the restriction of d^2 on $N \times N$ is C^∞ as required. □

Proof of Theorem 3.4.2. By Lemma A.2.1, under Assumption 3.4.1 and Assumption 3.4.2, the squared geodesic distances is smooth. The desired result is then obtained by applying Theorem 3.4.1. □

A.3 Proof of Theorem 3.4.4

Proof. The theorem is proved by combining Theorem 3.4.1 with Thm. 5 in Ciliberto et al. (2016). To characterize the constant $\mathsf{c}_\mathcal{L}$ we need an extra step.
Under Assumption 3.4.1 and Assumption 3.4.2 and the smoothness of $\mathcal{L}$, we can apply Theorem 3.4.1, which characterizes $\mathcal{L}$ as SELF. According to the proof of Theorem 3.4.1 and in particular of Lemma A.1.1, for any $y, z \in \mathcal{Y}$ we have

$$\mathcal{L}(y, z) = \langle \psi(y), V\psi(z) \rangle_\mathcal{H}$$

where $\mathcal{H} = H^2_s(\mathcal{M})$ with $s > d/2$, $\psi(y) = K_y(\cdot)$ where $K : \mathcal{M} \times \mathcal{M} \to \mathbb{R}$ is the reproducing kernel associated to $\mathcal{H}$ and $V : \mathcal{H} \to \mathcal{H}$ is the operator defined in Equation (A.1). In particular, by the isometry between the tensor space $\mathcal{H} \otimes \mathcal{H}$ and the space of Hilbert-Schmidt operators from $\mathcal{H}$ to $\mathcal{H}$, we have

$$\|V\|_{\mathrm{HS}} = \|\mathcal{L}\|_{\mathcal{H}\otimes\mathcal{H}}.$$

To conclude, since $\mathcal{L}$ is SELF, the following generalization bound in Thm. 5 from Ciliberto et al. (2016)

$$\mathcal{E}(\widehat{f}) - \mathcal{E}(f^*) \leq \|V\| \, \mathsf{q} \, \tau^2 \, n^{-\frac{1}{4}}$$

holds with probability at least $1 - 8e^{-\tau}$. Here, $\|V\|$ denotes the operator norm of $\|V\|$ and q is a constant depending only on $\mathcal{Y}$ and the distribution ρ (see end of proof of Lemma 18 for additional details). Finally, we recall, by the relation between the operator and Hilbert-Schmidt norm, that $\|V\| \leq \|V\|_{\text{HS}} = \|\mathcal{L}\|_{\mathcal{H}\otimes\mathcal{H}} = \mathsf{c}_{\mathcal{L}}$.

□